TECHNISCHE UNIVERSITÄT CHEMNITZ
FAKULTÄT FÜR MATHEMATIK

Large Series-Parallel Minors of Transitive Digraphs

An der Fakultät für Mathematik der Technischen Universität Chemnitz eingereichte

Diplomarbeit

von

Herrn Stefan Müller

geboren am 13. Februar 1986 in Karl-Marx-Stadt

Betreuender Hochschullehrer: Dr. F. Göring

Ausgabedatum: 19. Oktober 2011

Tag der Abgabe: 5. April 2012

Aufgabenstellung

Die Arbeit befasst sich mit folgender Problemstellung:

Ein transitiver Digraph sei gegeben. Einen series-parallelen Minor welcher Größe kann man finden? Probleme dieser Art treten im Zusammenhang mit Datenbanken auf.

Das Problem soll insbesondere für geeignete Kantengewichtungen betrachtet werden. Nach Sichtung der relevanten Literatur soll ein entsprechender Algorithmus zum Auffinden großer series-paralleler Minoren im Rahmen der Arbeit entwickelt und analysiert werden.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keinem anderen Prüfer als Prüfungsleistung eingereicht.

Chemnitz, 5. April 2012

Stefan Müller

Abstract

This work deals with the problem to find large series-parallel minors in transitive digraphs. In this context, different suitable edge weight functions for the given digraphs are discussed. To tackle this task, various algorithms are examined, both already known and specifically designed for this problem.

The relevance and strategies of related literature on similar problems and applications are studied and evaluated.

This issue arises in the context of reconstructing structures from multidimensional databases from relational data. Here, the required objects can be represented as series-parallel minors of certain digraphs, which hold the information about relationships between attributes of a database relation.

An important aspect of this work lies in the derivation and calculation of a specific possible edge weight function, that models the database structure well. Relevant properties of this weight function are discussed.

Zusammenfassung

Die Arbeit befasst sich mit der Bestimmung großer series-paralleler Minoren in transitiven Digraphen. Hierbei wird die Problemstellung insbesondere für verschiedene geeignete Kantengewichtungen betrachtet. Es werden sowohl bekannte als auch speziell für diese Aufgabe entworfene Algorithmen und Heuristiken diskutiert.

Dabei werden auch Arbeiten über verwandte Problemstellungen herangezogen und im Hinblick auf ihre Anwendbarkeit untersucht.

Die Aufgabenstellung ergibt sich im Zusammenhang mit der Rekonstruktion mehrdimensionaler Datenbankstrukturen aus relationalen Datenbanken. Die gesuchten Elemente können hier als series-parallele Minoren von Digraphen, die die Attributstruktur der relationalen Datenbank abbilden, aufgefasst werden.

Ein weiterer Aspekt der Arbeit besteht in der Herleitung und Konstruktion einer möglichen konkreten Kantengewichtung, die sich aus der Struktur der relationalen Datenbank ergibt. Relevante Eigenschaften dieser Gewichtung im gegebenen Kontext werden erörtert.

Acknowledgements

I wish to thank my supervisor Dr. Frank Göring for his competent support, his countless hints and all the constructive discussions. Furthermore, I wish to thank Prof. Dr. Christoph Helmberg for accepting the subject of this thesis and the people of the work group at the chair of Algorithmic and Discrete Mathematics for making me feel welcome. I also want to use this opportunity to thank Dr. Christian Kurze for introducing me to the topic from the perspective of database theory, which was the basis for this work.

I am also indebted to my family and many close friends who provided me with constant support, motivation and inspiration during the work on this thesis. Thank you.

Contents

1 Introduction

During the last years, multidimensional database systems have increasingly gained in importance in business intelligence applications [1]. Consequently, multidimensional concepts are very commonly used in practice, for example in data warehouse applications. However, quite often these concepts are not implemented as multidimensional databases, but instead are designed as relational databases [2, p. 352]. One of the reasons for this is that the relational model is well formalized at a conceptual level [3], whereas, on the other hand, there is still no concensus on mutidimensional data modeling [4]. This brought forth a variety of different relational database systems with a similar range of features and high interoperability. In addition, a lot of research effort has been put into relational database technology over the last decades, which led to great improvements in usability and performance of modern commercial tools for relational databases.

A main drawback of modeling multidimensional data as a relational database is that it is often not possible to represent the entire semantics inherent in those data. The explicite hierarchies, for example, which are incorporated in multidimensional databases, are often dismissed in favor of performance enhancements. However, such databases tend to be extremely large, expressive hierarchies become more and more important tools for database users for understanding the data. Likewise, hierarchies are also needed for meaningful data aggregation and it is not trivial to restore them if they are not available as part of the structure of the database [5].

The goal of this work is to find solutions to the hierarchy problem, i.e. reconstructing attribute hierarchies only from the data and structure of a single database relation. The main tool for the presented approaches is the FD-graph, a special transitive digraph that represents all the available information. This knowledge is retrieved by evaluating relationships between different attributes. It is shown that certain minors of this digraph can be regarded as hierarchies of the original relation. The hierarchy problem is therefore modeled as finding such minors in the FD-graph.

In order to assess how well these minors actually represent attribute hierarchies of the given relation, some kind of measure is needed. The here proposed idea is the introduction of an edge weight function for the FD-graph which rates the relevance of the attribute dependencies for any possible hierarchy. It is not obvious how such a weight function has to be modeled and what its characteristics should be. Therefore, it is not only important to discuss the problem for abstract weight functions with certain special properties, but also to give close attention to the explicite design of a particular weight function that is able to express the above mentioned attribute dependencies properly.

The focus of the following considerations lies on series-parallel minors, which correspond to generalized hierarchies, including also simple and multiple independent hierarchies as special cases. These are the most relevant attribute hierarchies occuring in practice. Both well-known and specifically designed algorithmic approaches to find series-parallel minors with large aggregate weight are discussed.

In chapter 2 necessary terms and notations from graph theory and database theory are introduced. A formal model of the hierarchy problem that introduces the FD-graph is developed in chapter 3 together with methods for cycle-removal and a way for reducing the problem to a subclass of the FD-graphs. Chapter 4 discusses a paper by Bein et al. [6], which examines a related issue. Solutions for two special cases of weight functions are deduced in chapter 5, completed by an evaluation of the Greedy Algorithm for this problem with general weight functions. A special weight function is motivated and derived in chapter 6. Finally, chapter 7 concludes this work and summarizes results and open problems.

2 Preliminaries

This chapter contains most of the definitions that are the basis for the presented ideas, their formal analysis and the argumentations. After some basic notations, there is a section introducing general terms from graph theory, followed by section 2.2 focussing solely on series-parallel digraphs because of their special relevance to this work. Section 2.3 deals with concepts from relational database theory and section 2.4 introduces attribute hierarchies, which are fundamental structures in multidimensional databases.

For the cardinality of a set S the notation $|S|$ is used. The *power set* of a set S, i.e. the set of all subsets of S, is denoted by 2^S. A *partition* of S is a non-empty set $\mathcal{P} = \{P_1, \ldots P_k\}$ such that $P_i \neq \emptyset$ and $P_i \cap P_j = \emptyset$ for all $i, j \in \{1, \ldots k\}$ with $i \neq j$ and such that

$$\bigcup_{i=1}^{k} P_i = S.$$

Note that a partition of S is a subset of the power set of S.

The set of natural numbers is denoted by $\mathbb{N}$, the set of real numbers by $\mathbb{R}$ and the positive real numbers together with 0 by $\mathbb{R}_+$.

The *falling factorial* $x^{\underline{n}}$ is defined by

$$x^{\underline{n}} = x(x-1)\ldots(x-n+1)$$

for $x \in \mathbb{R}$ and $n \in \mathbb{N}$ [7, p. 7].

Let S be a non-empty set with a weight function $c : 2^S \to \mathbb{R}_+$. A *property over* S is a set $\mathcal{P} \subseteq 2^S$. An element $A \in \mathcal{P}$ is *maximal* w.r.t. $\mathcal{P}$ if there is no $B \in \mathcal{P}$ such that $A \subset B$. An element $A \in \mathcal{P}$ is *maximum* w.r.t. $\mathcal{P}$ if for every $B \in \mathcal{P}$ it holds that $c(A) \geq c(B)$. If no weight function is provided, it is implicitly assumed that $c(A) = |A|$ for all $A \in \mathcal{P}$. Conversely, an element $A \in \mathcal{P}$ is *minimal* w.r.t. $\mathcal{P}$ if there is no $B \in \mathcal{P}$ such that $B \subset A$. An element $A \in \mathcal{P}$ is *minimum* w.r.t. $\mathcal{P}$ if for every $B \in \mathcal{P}$ it holds that $c(A) \leq c(B)$. An *optimal* element of $\mathcal{P}$ denotes either a maximal or a minimal element, depending on whether a maximization or a minimization problem is considered, respectively. Analogously, an *optimum* element is either maximum or minimum, depending on the context.

The terms from the previous paragraph are used rather informally throughout this work without explicitly indicating S and $\mathcal{P}$, e.g. a maximal matching in a digraph G denotes a maximal element w.r.t. $\mathcal{P}$, where $\mathcal{P}$ is just the set of all matchings in G, and S is the edge set of G.

2.1 Graph Theory

This work follows mostly the definitions from [8], but some notations from [9] and [10] are also used.

A *digraph* is an ordered pair $G = (V, E)$ where V is a set of *vertices* and E is a set of *edges* together with two incidence functions $\text{init} : E \to V$ and $\text{ter} : E \to V$, assigning to every edge e an *initial vertex* $\text{init}(e)$ and a *terminal vertex* $\text{ter}(e)$, its two *end-vertices*. The *vertex set* and *edge set* of G is also denoted by $V(G)$ and $E(G)$, respectively. In the following it is always tacitly assumed that vertex set and edge set are disjoint. Edges in E and vertices in V are said to be *contained in G* or just *in G*. The digraph $(\emptyset, \emptyset)$ is called *empty*; if at least $V \neq \emptyset$ then G is *non-empty*.

Edges with the same initial and the same terminal vertex are called *multiple*. An edge whose end-vertices are identical is called a *loop*. This work does not permit multiple edges and loops in a digraph unless stated otherwise. Therefore it can be stated that $E \subset V \times V$ and an edge e can be written as an ordered pair $(\text{init}(e), \text{ter}(e))$ of two different vertices.

A digraph is commonly depicted by drawing dots for its vertices and arrows between those dots representing the edges. Here, the actual positions of the dots are irrelevant; only the incidence relations between the pictured vertices and edges are of interest [9, p. 2]. An example of a digraph with 11 vertices and 13 edges can be seen in figure 2.1. The vertices and edges are labeled with v_i and e_j, respectively.

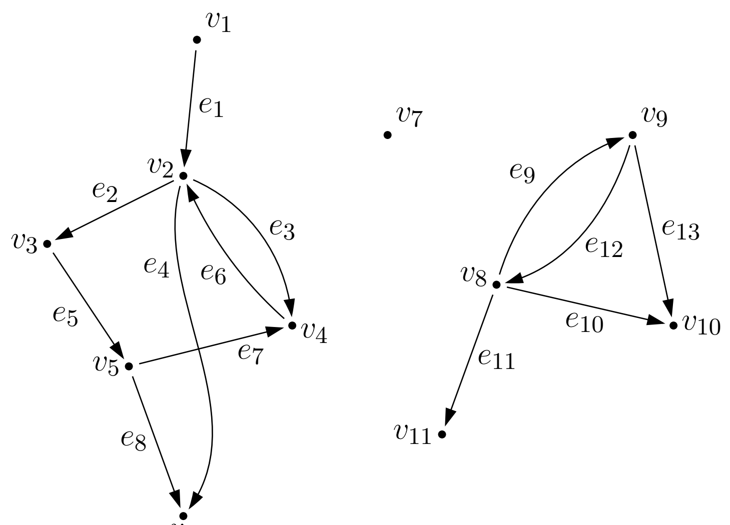

Figure 2.1 A digraph

The end-vertices of an edge $e = (v, w) \in E$ are said to be *incident* to e and e is an edge *from v to w* or *between v and w*. The initial vertex v of e

is an *in-neighbor* of the terminal vertex w and conversely w is *out-neighbor* of v. The edge e *leaves* v and *enters* w. Two vertices in G are *adjacent*, or *neighbors*, if there is an edge between them in G. Two edges are *adjacent* if they share a common end-vertex. The edge (w, v) is the *opposite edge* of e.

The sets $N_G^-(v)$ and $N_G^+(v)$, defined by

$$N_G^-(v) = \{u \in V \setminus \{v\} : (u, v) \in E\},$$
$$N_G^+(v) = \{u \in V \setminus \{v\} : (v, u) \in E\},$$

are the *in-neighborhood* and *out-neighborhood* of v in G, respectively. The *neighborhood* of v in G is $N_G(v) = N_G^-(v) \cup N_G^+(v)$. These neighborhoods can analogously be defined for sets of vertices. For a set $U \subseteq V$ let

$$N_G^-(U) = \bigcup_{w \in U} N_G^-(w) \setminus U, \quad N_G^+(U) = \bigcup_{w \in U} N_G^+(w) \setminus U$$

and $N_G(U) = N_G^-(U) \cup N_G^+(U)$. Here and in following definitions, the indices specifying the considered graph may be omitted if no ambiguities arise.

Let $E_G^-(v) = \{(w, v) \in E\}$ and $E_G^+(v) = \{(v, w) \in E\}$ denote the sets of edges incident to v. Note that because of the exclusion of parallel edges and loops it holds that $|E_G^-(v)| = |N_G^-(v)|$ and $|E_G^+(v)| = |N_G^+(v)|$.

The *in-degree* $d_G^-(v) = |E_G^-(v)|$ of a vertex $v \in V$ in G is the number of edges in E with v as their terminal vertex and the *out-degree* $d_G^+(v) = |E_G^+(v)|$ of v in G is the number of edges with v as their initial vertex. The *degree* of v in G is $d_G(v) = d_G^-(v) + d_G^+(v)$. A vertex $v \in V$ is called a *subdividing vertex* if $d_G^-(v) = d_G^+(v) = 1$. Furthermore, $v \in V$ is a source in G if $d_G^-(v) = 0$ and $d_G^+(v) > 0$ and contrarily, it is a sink in G if $d_G^-(v) > 0$ and $d_G^+(v) = 0$. A digraph is *two-terminal* if it has exactly one source and exactly one sink.

Example 2.1 The vertex v_3 of the digraph in figure 2.1 is a subdividing vertex. Examples for the degrees of vertices are $d^-(v_2) = 2$ and $d^+(v_2) = 3$. The in-neighborhood of v_2 is $N^-(v_2) = \{v_1, v_4\}$. The in-neighborhood of $U = \{v_1, v_2, v_6\}$ is $N^-(U) = \{v_4, v_5\}$. The only source is v_1 and the digraph has three sinks, namely v_6, v_{10} and v_{11}.

Let $G' = (V', E')$ be another graph. If $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$ then G' is a *subdigraph* of G and G is a *superdigraph* of G', written as $G' \subseteq G$. In this case G *contains* G' or G' is *in* G. If $E' = E \cap (V' \times V')$, then G' is called an *induced* subdigraph of G, denoted by $G' = G[V']$. In this case, V' is said to *induce* G' in G. A subdigraph G' of G is *spanning* if $V' = V$.

For a given set of vertices U, the digraph $G[V \setminus U]$ may be denoted by $G - U$, i.e. $G - U$ results from G by deleting all vertices in $U \cap V$ and all their incident edges from G. The term $G - V'$ may also be written as $G - G'$ instead. If $U = \{u\}$ is a singleton then $G - \{u\}$ may be abbreviated to $G - u$. For a given set of edges $F \subset V \times V$ the term $G - F$ denotes the digraph $(V, E \setminus F)$ and $G + F$ is the graph $(V, E \cup F)$. Analogously, $G - \{f\}$ and $G + \{f\}$ may be shortened to $G - f$ and $G + f$, respectively, if this does not lead to ambiguities. The *union* $G \cup G'$ of G and G' is the graph $(V \cup V', E \cup E')$.

Let P be a non-empty digraph with $V(P) = \{v_0, \ldots, v_n\}$, $v_i \neq v_j$ for all $i, j \in \{0, \ldots, n\}$ with $i \neq j$, and $E(P) = \{e_1, \ldots, e_n\}$. If $v_{i-1} = \text{init}(e_i)$ and $v_i = \text{ter}(e_i)$ for all $i \in \{1, \ldots, n\}$ then P is a *(directed) path*. The vertex v_0 is the *initial vertex* and v_n is the *terminal vertex* of P and P is said to be a path *from v_0 to v_n* or a *v_0-v_n-path*. The edges and vertices of P *lie on* P and the *length* of P is the number of its edges $|E(P)| = n \geq 0$.

The *concatenation* of a v-x-path P_1 and an x-w-path P_2 which have only x in common is the v-w-path $P_1 \cup P_2$, abbreviated by P_1P_2. If P is a v-w-path with $x, y \in V(P)$, then the notations xP, Py and xPy denote the unique x-w-path, v-y-path and x-y-path in P, respectively.

An *undirected path* P' results from a path P by replacing any number of edges $(v, w) \in E(P)$ by the inverse edge (w, v). The end-vertices of P' are said to be *linked* by P'.

A digraph G is *connected* if for every pair of vertices in G there is an undirected path in G that links them. The maximal connected subdigraphs of G are called *components* of G. A vertex set $S \subset V$ with $|S| = k$ is called a *k-separator* in G if G is connected and $G - S$ is not. A digraph G is *k-connected* if $|V| > k$ and there is no $k - 1$-separator in G. The maximal k-connected subdigraphs of G are called *k-connected components* of G.

Example 2.2 The digraph in figure 2.1 contains the path

$$P = (\{v_1, v_2, v_3, v_5, v_6\}, \{e_1, e_2, e_5, e_8\}).$$

However, the depicted digraph is not connected, because there is, for example, no undirected path between v_1 and v_8; the subdigraph induced by $\{v_1, \ldots, v_6\}$ is connected and also a component of this digraph. Let $P' = (\{v_5, v_4\}, \{e_7\})$. The concatenation of v_3Pv_5 and P' is the path $v_3Pv_5P' = (\{v_3, v_5, v_4\}, \{e_5, e_7\})$.

A *cycle* C is a digraph $P + e$, where P is a path and e is an edge from the terminal vertex of P to its initial vertex. The *length* of C is defined as the number of its edges. Note that because of the exclusion of loops, cycles have at least length 2. A digraph is *cyclic* if it contains a cycle, otherwise it is *acyclic*.

An s-t-DAG G' is a two-terminal acyclic digraph with source s and sink t, $s \neq t$. It is easy to see that for every vertex $v \in V(G')$ there is an s-t-path that contains v, otherwise s and t would not be unique source and sink, respectively.

An edge $(v, w) \in E$ is *transitive* in G if there is a v-w-path in G not containing (v, w). If G does not contain any transitive edges it is *intransitive*. If for every v-w-path in G of length at least 1 the edge (v, w) is also in G, then G is *transitive*. The *transitive closure* of a digraph G is the digraph T with $V(T) = V$ that contains the edge (v, w) for $v, w \in V$, $v \neq w$ if and only if there is a v-w-path in G.

A *transitive reduction* R of G is an intransitive subdigraph of G such that the transitive closures of R and G coincide. The transitive reduction is unique if G is acyclic [8, pp. 37f.], therefore it is possible to denote the transitive reduction for an acyclic digraph G by $TR(G)$.

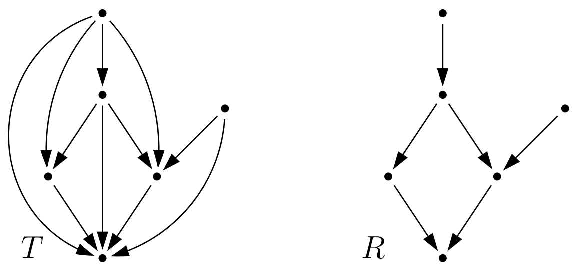

Figure 2.2 A transitive digraph T and an intransitive digraph R

Example 2.3 Figure 2.2 depicts a transitive digraph T and an intransitive digraph R. In addition, T is the transitive closure of R and R is the transitive reduction of T.

Let $e \in E$ be an edge in G. By G/e the digraph that results from the *contraction* of $e = (v, w)$ to a new vertex $v_e \notin V \cup E$ is denoted. It is defined by $G/e = (V', E')$ with $V' = (V \setminus \{v, w\}) \cup \{v_e\}$ and

$$E' = \big\{(x, y) \in E : \{v, w\} \cap \{x, y\} = \emptyset\big\} \cup \big\{(v_e, x) : x \in N_G^+(\{v, w\})\big\} \\ \cup \big\{(x, v_e) : x \in N_G^-(\{v, w\})\big\}.$$

A digraph that is obtained from G by a series of vertex deletions, edge deletions and contractions of edges is a *minor* of G [9, pp. 18–20].

A *matching* $M \subseteq E$ in G is a set of pairwise not adjacent edges. A set of vertices $U \subset V$ is *matched* by M if all $v \in U$ are incident to an edge in M. In this case M is said to be a matching *of* U.

A *vertex cover* $S \subseteq V$ of a digraph is a subset of its vertices, such that every edge of the digraph is incident to at least one vertex in S.

The *complement* of the digraph G, denoted by $\overline{G}$, is defined by $V(\overline{G}) = V$ and $E(\overline{G}) = \{(v, w) : v, w \in V, v \neq w, (v, w) \notin E\}$.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are called *isomorphic* if there exists a bijective function $\varphi : V \to V'$ such that $(v, w) \in E$ if and only if $(\varphi(v), \varphi(w)) \in E'$ for all $v, w \in V$, $v \neq w$.

The digraph G is *complete* or a *clique* if it contains all possible edges, i.e. if $E = \{(v, w) : v, w \in V, v \neq w\}$. For a given number n of vertices the complete digraph is unique up to isomorphisms, therefore this digraph can be denoted by K^n. The digraph consisting of exactly two vertices and one edge between them is denoted by $\vec{K}^2$.

2.2 Series-Parallel Digraphs

This section introduces the class of series-parallel digraphs and some basic properties of those digraphs.

A *series-parallel* digraph, sometimes called edge or arc series-parallel [6, 8] in contrast to vertex series-parallel digraphs, can be recursively defined as follows [8, p. 48]. The digraph $\vec{K}^2$ is series-parallel. Let G_1 and G_2 be two series-parallel digraphs with $V(G_1) \cap V(G_2) = \emptyset$. Then the digraphs obtained by any of the following operations are also series-parallel.

- *Parallel composition*: Choose a source s_i and a sink t_i from G_i for $i = 1, 2$. Identify s_1 with s_2 and t_1 with t_2.
- *Series composition*: Choose a sink t from G_1 and a source s from G_2 and identify t with s.

Note that the parallel composition of two $\vec{K}^2$ is again $\vec{K}^2$ because multiple edges are not allowed in the here presented context.

A series-parallel digraph is two-terminal because the two different compositions keep the number of sources and sinks constant [8]. These digraphs are also acyclic, as the operations do not create cycles [11]. An example for a series-parallel digraph is shown in figure 2.3.

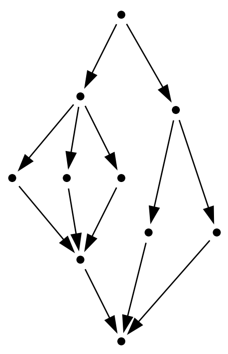

Figure 2.3 A series-parallel digraph

Observation 2.4 A series-parallel digraph G that is not isomorphic to $\vec{K}^2$ always contains a subdividing vertex.

Proof. Since G is not isomorphic to $\vec{K}^2$ it has at least two edges. If $|E(G)| = 2$ then G is a series composition of two $\vec{K}^2$, and the vertex in G that is neither sink nor source is a subdividing vertex. Now let $|E(G)| = k \geq 3$ and assume that the claim is true for all G' with $|E(G')| < k$. Clearly, G is the result of either a parallel composition or a series composition of two series-parallel digraphs, say G_1 and G_2, of which at least one is not isomorphic to $\vec{K}^2$; w.l.o.g. let G_1 be that digraph. By induction hypothesis, G_1 has a subdividing vertex v. The two compositions only change the degrees of source and sink of G_1 and G_2, so v is still a subdividing vertex in G. □

The following definition, which has been shown to be equivalent to the above in [11, lemma 2, p. 302], is based on the notion of parallel and series reduction which can in this form also be found in [6, p. 3].

- *Parallel reduction*: Replace any number of edges $e_1, \ldots, e_k$ from v to w with a single edge (v, w).
- *Series reduction*: Replace two edges (u, v) and (v, w), with v being a subdividing vertex, by the edge (u, w) and remove v.

Now a digraph G is called series-parallel if it can be turned into the digraph $\vec{K}^2$ by a sequence of parallel and series reductions. Note that the intermediate result of any of those reductions may contain multiple edges. The reductions have the Church-Rosser property, i.e. the order of those reductions in the sequence does not influence the final outcome [6, p. 3].

It is also possible to characterize series-parallel digraphs by means of the forbidden minors [11, p. 311] [12], a characterization that receives special attention in chapter 4, as proposition 2.5 shows. The *interdictive digraph*, or IG, is the s-t-DAG with vertices $\{s, v, w, t\}$ and edges $\{(s, v), (s, w), (v, w), (v, t), (w, t)\}$. It is depicted in figure 2.4.

Proposition 2.5 A two-terminal acyclic digraph is series-parallel if and only if it does not contain an IG as minor.

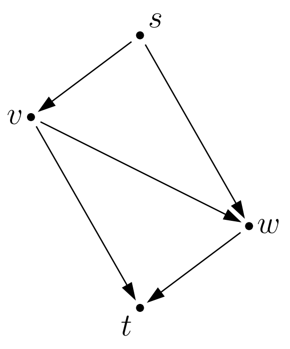

Figure 2.4 The interdictive digraph

2.3 The Relational Data Model

This section introduces some terms and notations from relational database theory adapted from definitions by Levene and Loizou [5, pp. 28f.] and by Lausen [13, pp. 44f.].

Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a non-empty, finite set of non-empty, pairwise disjoint sets. The *domain* of $\mathcal{S}$ is defined as

$$\operatorname{dom}\mathcal{S} = \big\{\{s_1, \ldots, s_n\} : s_i \in S_i \ \forall i \in \{1, \ldots, n\}\big\}.$$

The elements of $\operatorname{dom}\mathcal{S}$ are well defined sets with cardinality n because of the elements of $\mathcal{S}$ being disjoint.

A *relation schema* $R = \{A_1, \ldots, A_m\}$, $m \geq 1$, is a non-empty, finite set of non-empty, finite, pairwise disjoint sets, called *attributes*[1]. An *R-tuple* is a member of $\operatorname{dom} R$. A *relation*[2] r over R is a set of R-tuples, thus $r \subseteq \operatorname{dom} R$. Note that any two R-tuples in r are distinct and that r is finite because $\operatorname{dom} R$ is finite.

Let τ be an R-tuple and let $X \subseteq R$ be a set of attributes. The *projection* of τ onto X is the set $\tau[X] \subseteq \tau$ defined as $\tau[X] = \{\tau \cap A : A \in X\}$. It obviously holds that $\tau[R] = \tau$.

A *key* $k = (R, X)$, $X \subseteq R$ and $X \neq \emptyset$, over R is an ordered pair of a relation schema R and a set of attributes X in R. A relation r over R *satisfies* k if $\forall \tau, \upsilon \in r$ it holds that

$$\tau[X] = \upsilon[X] \Rightarrow \tau = \upsilon.$$

Following [5, definition 2.5, p. 229], a *functional dependency* over the relation schema R is a statement of the form $X \to Y$, where $X \cup Y \subseteq R$. The functional dependency is said to be *trivial* if $Y \subseteq X$. A functional dependency is *satisfied* by the relation r over R if $\forall \tau, \upsilon \in r$ it holds that

$$\tau[X] = \upsilon[X] \Rightarrow \tau[Y] = \upsilon[Y].$$

Satisfiability of functional dependencies is reflexive, i.e. r always satisfies $X \to X$, and transitive, i.e. if r satisfies the functional dependencies $X \to Y$ and $Y \to Z$ then it also satisfies $X \to Z$.

2.4 Hierarchies in Relations

A Hierarchy is a concept known from the theory of multidimensional databases [4, 14, 15] used to group multidimensional data. It is not very common to introduce hierarchies in the context of relational databases, but it can be done without further problems and is very useful for the purpose of this work, as it eliminates the necessity to introduce the full theory of multidimensional databases.

The following definitions are mostly adapted from basic ideas of the works by Abelló et al. [16, 17], in which an extensible conceptual model for multidimensional databases was developed.

Let R be a relation schema. A *hierarchy* over R is an intransitive two-terminal acyclic digraph $H = (V, E)$ with source $\bot_H \in V$ and sink $\top_H \in V$, a notation proposed in [14, p. 394], where $V \setminus \{\top_H\}$ is a partition of R. The vertices of H are called *hierarchy levels*, $\bot_H$ is the *atomic hierarchy level* of H [16, p. 4-5] and $\top_H$ is the *top hierarchy level* of H. A relation r over R is said to *satisfy* H if for every edge $(v, w) \in E$ the functional dependency $v \to w$ is satisfied by r.

The reason for defining hierarchies as intransitive is a natural consequence of the properties of functional dependencies. Transitive edges don't hold any additional information for the hierarchy, because the satisfiability of the functional dependencies they represent already completely depends on other edges.

Hierarchies can be classified and categorized based on their structure. The following hierarchy types are derived from [2, pp. 353–366]. Not all proposed hierarchies are presented here because some of them are not relevant in the context of this work.

A hierarchy H is *simple* if it is a $\bot_H$-$\top_H$-path. A hierarchy H is *independent multiple* if it is the union of more than one $\bot_H$-$\top_H$-paths that are pairwise vertex-disjoint besides $\bot_H$ and $\top_H$. A series-parallel hierarchy that is neither simple nor independent multiple is called *generalized*. Finally, if a hierarchy is none of the three aforementioned types, then it is *complex*.

3 The Hierarchy Problem

The first main question is how the problem of constructing a hierarchy H, which is satisfied by a given relation r over a relation schema R, can be formulated as a graph minor problem. A digraph which has a hierarchy over R as a minor should have two properties. First, its set of vertices should contain a different vertex for every attribute in R, as to allow any granularity in H, and second, it should model as much information about the functional dependencies satisfied by r as possible, providing data for the edges of H.

These requirements are met by the FD-graph, which is introduced in section 3.1 together with the formal objectives of this work. In general, this FD-graph may be cyclic, so section 3.2 shows approaches to cycle elimination. In section 3.3 it is demonstrated that, for general acyclic FD-graphs, the given problem can be reduced to the same problem for 3-connected acyclic FD-graphs.

3.1 The FD-Graph

Let r be a relation over R. The *FD-graph* of r is a digraph $G = (V, E)$, where $V = 2^R$ and $E = \{(v, w) : v, w \in V \wedge r \text{ satisfies } v \to w\}$, together with a weight function $c : E \to \mathbb{R}_+$ called the *information content* of e. A similar idea has been used in [18, definition 5.5, p. 426]. The actual values of c are discussed in chapter 6.

It is easy to see that G is connected and has a unique sink $\emptyset \in V$ and at most one source $R \in V$. If G is acyclic then $R \in V$ is indeed a unique source of G. In addition, G is transitive because satisfiability of functional dependencies is transitive, as discussed in section 2.3.

It is clear from the definition of the FD-graph that it is not useful to add vertices or edges to the FD-graph to obtain a hierarchy. The preliminary goal can thus be formulated as follows.

Objective 3.1 For a given relation r over R, find a minor M of the FD-graph of r with maximum information content, such that

(1) M is acyclic two-terminal and
(2) the set of vertices $V(M) \setminus \{\emptyset\}$ is a partition of R.

So far, it is not clear what the information content of a minor M is, which is very problematic. It is not obvious how the information content of edges created by contractions could be derived or computed based on the information content of edges in E. Hence, it would be necessary to extend c explicitly to take arguments from $2^V \times 2^V$ rather than E, as sets of vertices of G correspond to vertices in a minor of G.

Contracting the edge (v, w), however, can be conceived as merging its two end-vertices. In the given context, it is therefore reasonable to identify the new vertex that emerges from the contraction with the vertex $v \cup w$, which is already in V. This operation may create new edges, namely edges from $N^-(w) \setminus N^-(v)$ to $v \cup w$, if $(w, v) \notin E$, for which no justification in terms of functional dependencies satisfied by r exist because they are not in E. Therefore, instead of contracting (v, w), it is a better option to delete the vertices v and w, which results in the same digraph but without any additional edges.

These observations motivate the consideration of subdigraphs of G instead of minors of G as candidates for a hierarchy over R. Extending the information content to subdigraphs of G is not hard. Recall that hierarchies are defined as intransitive. With this in mind, the *information content of subdigraph M* of the FD-graph can be defined by

$$c(M) = \begin{cases} \sum\limits_{e \in E(TR(M))} c(e) & \text{if } M \text{ is acyclic,} \\ -\infty & \text{otherwise.} \end{cases}$$

Hence, the following adjustion to objective 3.1 can be made.

Objective 3.2 For a given relation r over R, find a subdigraph M of the FD-graph of r with maximum information content $c(M)$, such that

(1) M is acyclic two-terminal and
(2) the set of vertices $V(M) \setminus \{\emptyset\}$ is a partition of R.

Note that property (1) is equivalent to M being an arbitrary hierarchy. More relevant than objective 3.2 is the following special case. It restricts the possible solutions M to hierarchies that are not complex. Note that a complex hierarchy is very general and not desirable in practice.

Objective 3.3 For a given relation r over R, find a subdigraph M of the FD-graph of r with maximum information content $c(M)$, such that

(1) M is series-parallel and
(2) the set of vertices $V(M) \setminus \{\emptyset\}$ is a partition of R.

The *objective function σ* can now be defined for any digraph G with the information content c as above, by $\sigma(G) := c(M)$, with M being a subdigraph of G with maximum information content satisfying objective 3.3.

Section 3.2 deals with the question of achieving (1) from objective 3.2. Finding a large series-parallel subdigraph in terms of c, i.e. objective 3.3 (1), is considered in chapter 5. Discussion of (2) is postponed.

3.2 Dealing with Cycles

This section discusses how all cycles can be removed from the FD-graph G, while preserving unique source and unique sink. Hence, the result of this operation is a two-terminal acyclic subdigraph of G. Working with an acyclic digraph simplifies the considerations in section 3.3 and in chapter 5, but it is easy to see that this approach also excludes some feasible solutions for objective 3.3.

Naturally, the following methods aim for a digraph with high information content. However, the weight of any digraph G' in this section always denotes the sum of the weights of all edges in $E(G')$, rather than only those edges in the transitive reduction of G', i.e.

$$c(G') = \sum_{e \in E(G')} c(e).$$

The main reason for this definition is that mostly standard approaches are presented here, which work only with this type of weight function. In addition, the properties of the weight function c have no influence on the structure of the resulting digraph, since a maximal acyclic subdigraph of a complete digraph is unique up to isomorphisms, i.e. there is only one maximal acyclic digraph on a given number of vertices.

Note that because the FD-graph G is transitive, the vertices of every cycle in G induce a clique in G. Thus, the above stated problem can be reformulated as finding all maximal cliques contained in the FD-graph and constructing maximal acyclic subdigraphs in all those cliques. This is outlined in the following algorithm.

Algorithm 3.4 (Acyclicity Algorithm)

```
Input:  An FD-graph $G$
Output: A maximal acyclic subdigraph $G'$ of $G$
1. Set $G' := G$.
2. Find a maximal complete subdigraph $K$ of $G'$.
3. If $|V(K)| > 1$ then compute a maximal acyclic subdigraph $H$ of $K$ and go
   to 4. Otherwise stop.
4. Remove all edges in $E(K) \setminus E(H)$ from $E(G')$ and go to 2.
```

Essentially, this algorithm searches for maximal cycles in G and replaces them by acyclic digraphs with the same number of vertices. It is clear that this algorithm works correctly because it does not add new edges and step 3 removes edges until there are no more cycles. The subdigraph G' is also maximal acyclic because H in step 3 is maximal acyclic. In the here presented case of a transitive input digraph G, the algorithm already incorporates the preparation step suggested by Berger and Shor [19, remark 1, p. 238], which prevents the removal of edges not in any cycle.

Before discussing the important steps of the algorithm in detail, some observations shall be made. They show that the output of algorithm 3.4 satisfies property (1) of objective 3.2.

Observation 3.5 The result of algorithm 3.4 is a transitive digraph.

Proof. Removing edges in step 4 does not destroy transitivity because H from step 3 is maximal acyclic and therefore also transitive. □

Observation 3.6 Let H be a maximal acyclic subdigraph of a complete digraph. Then H has exactly one source and one sink.

Proof. Because of the acyclicity of H it contains at least one source and one sink. Now assume that there are two sources $s_1, s_2 \in V(H)$. There is no s_1-s_2-path in H, so adding (s_2, s_1) to $E(H)$ does not create a cycle in H. This contradicts the maximality of H, so H cannot have two sources. A similar argumentation shows that H has at most one sink. □

Observation 3.7 The digraph G' constructed by algorithm 3.4 has unique source and unique sink.

Proof. Let the FD-graph G be the input of the algorithm. It was already stated that, if G is acyclic, then it is also two-terminal and the algorithm simply returns $G' = G$. Otherwise, G has a unique sink $t = \emptyset$ that cannot be part of any K with $|V(K)| > 1$ in step 2 of algorithm 3.4. By observation 3.6, a unique sink t' in H is created in step 3, but t' is not a sink in G' because the edge $(t', \emptyset)$ is in G and never removed by the algorithm.

The argument is the same for the existence of a unique source if G already has a source, which would be R. However, if G has no source, then there is a cycle in G that contains R. At some iteration of the algorithm, this cycle is part of K in step 2 with $|V(K)| > 1$. Again, by observation 3.6, step 3 produces a new source s in H. After step 4, s is also a source in G' because, if there was an edge (v, s) in $E(G')$ with $v \notin V(K)$, there would also be the edges $(s, R) \in E(K)$ and $(R, v) \in E(G')$, together forming a cycle in G', which contradicts $v \notin V(K)$. □

Step 2 of algorithm 3.4 can be done by first searching for a complete subdigraph with two vertices and then successively increasing it by finding vertices that are both in- and out-neighbors of one of the vertices of the so constructed subdigraph. This method is correct because of the transitivity of the FD-graph. The algorithm stops if no complete subdigraph with at least two vertices can be found in G', which is true if and only if G' is acyclic. Because of the transitivity of G and because no edge has to be considered twice, the running time of step 2 is $O(|E|)$ for all iterations of algorithm 3.4, given an algorithmic representation of G that allows to check for an opposite edge in constant time.

In step 3 the best solution for H would be a maximum acyclic subdigraph of K. This problem, which is equivalent to finding a maximum linear ordering of $V(K)$, has unfortunately proven to be NP-hard [20, p. 113] in the general case, even for unweighted digraphs [21–23]. There is also no efficient α-approximation algorithm known with $\alpha > \frac{1}{2}$ and it is unlikely that there is one [21].

A simple and efficient approach to step 3 is the Greedy Algorithm. This heuristic is able to find optimum solutions for optimization problems that have matroid structure, e.g. finding the minimum spanning tree in an undirected graph [8, p. 342], and has also proven to be useful as an approximation algorithm for many independence systems. Independence systems and matroids are introduced on the following pages. Algorithm 3.8 shows the Best-In-Greedy Algorithm, derived from [24, p. 59], that creates a maximal solution by adding edges with decreasing weight to an empty digraph.

Algorithm 3.8 (Best-In-Greedy Algorithm)

```
Input:  A complete digraph $K$ with weights $c : E(K) \to \mathbb{R}_+$
Output: A maximal acyclic subdigraph $H$ of $K$
1. Set $X_0 := \emptyset$ and $j := 0$.
2. If $E(K) \setminus X_j$ contains an element $e$ such that $X_j \cup \{e\}$ does not form a cycle
   in $K$, then choose such an element $e_{j+1}$ with maximum weight and go to 3,
   otherwise set $H := (V(K), X_j)$ and stop.
3. Set $X_{j+1} := X_j \cup \{e_{j+1}\}$ and increase $j$ by 1. Go to 2.
```

Again, it is obvious that the algorithm correctly yields a maximal acyclic subdigraph of K. It considers every edge once and has to decide whether this edge forms a cycle, e.g. by a depth-first search [25, chapter 3], therefore it has running time at most $O(|E(K)| \cdot (|E(K)| + |V(K)|)) = O(|E(K)|^2)$ if the edges are already sorted. For further analysis of this algorithm some notions from matroid theory are required. These definitions are taken from [26, chapter 13].

Let E be a finite set, the *ground set*, and let $\mathcal{F} \subseteq 2^E$ be a collection of subsets of E [24, p. 7]. The set system $(E, \mathcal{F})$ is an *independence system* if the following two conditions hold.

(M1) $\emptyset \in \mathcal{F}$;

(M2) If $X \in \mathcal{F}$ and $Y \subseteq X$ then $Y \in \mathcal{F}$.

The elements of $\mathcal{F}$ are called *independent* and the elements of $2^E \setminus \mathcal{F}$ are called *dependent*. Minimal dependent sets are called *circuits* and maximal independent sets *bases*. Maximal independent subsets of $X \subseteq E$ are called *bases of* X.

An independence system $(E, \mathcal{F})$ is a *matroid* if

(M3) For all $X, Y \in \mathcal{F}$ with $|X| > |Y|$ there is an $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{F}$.

Now consider $E = E(K)$ and $\mathcal{F} = \{X \subseteq E : (V(K), X) \text{ is acyclic}\}$. It is easy to see that $(E, \mathcal{F})$ is an independence system. To emphasize the underlying digraph, this independence system is denoted by $(E, \mathcal{F})_K$. Together with $F = E(H)$, input and output of algorithm 3.8 can be formulated as follows.

Algorithm 3.9 (Best-In-Greedy Algorithm [26, p. 303])
Input: An independence system $(E, \mathcal{F})_K$ and weights $c : E \to \mathbb{R}_+$
Output: A basis F of $(E, \mathcal{F})$

Algorithm 3.9 is an algorithm for the maximization problem for $(E, \mathcal{F}, c)$ [26, p. 303]. The following well-known theorem, which is provided without proof, reveals the connection between independence systems and the Greedy Algorithm.

Theorem 3.10 (Edmonds-Rado Theorem [26, theorem 13.20, p. 306])
An independence system $(E, \mathcal{F})$ is a matroid if and only if the Best-In-Greedy Algorithm finds an optimum solution for the maximization problem for $(E, \mathcal{F}, c)$ for all weight functions $c : E \to \mathbb{R}_+$.

Unfortunately, $(E, \mathcal{F})_K$ is not guaranteed to be a matroid for $|V(K)| > 3$ in the general case, as the following example shows. Let $V(K) = \{v_1, v_2, v_3, v_4\}$,

$$X = \{(v_2, v_3), (v_3, v_1), (v_3, v_4), (v_4, v_1)\} \subseteq E \quad \text{and}$$
$$Y = \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\} \subseteq E$$

as depicted in figure 3.1. Here, X and Y are independent sets with $|X| > |Y|$, but Y is already maximal independent in $X \cup Y$, which violates (M3). So in general the Greedy Algorithm does not compute the optimum solution by theorem 3.10. The question remains whether there is at least a lower bound for the approximation quality of the Greedy Algorithm. For this, some more definitions from [26] are needed.

Let $(E, \mathcal{F})$ be an independence system. For $X \subset E$ the *rank* of X is defined by

$$r(X) := \max\{|Y| : Y \subseteq X, Y \in \mathcal{F}\}$$

and the *lower rank* of X is defined by

$$\rho(X) := \min\{|Y| : Y \subseteq X, Y \in \mathcal{F} \text{ and } Y \cup \{x\} \notin \mathcal{F}\ \forall x \in X \setminus Y\}.$$

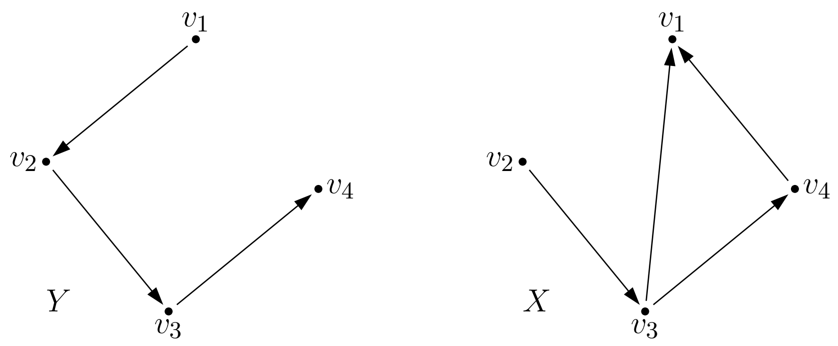

Figure 3.1 Two independent elements of $(E, \mathcal{F})_{K^4}$ violating (M3)

Thus the rank of X is the size of the largest basis in X and the lower rank of X is the size of the smallest basis in X. Calculated from rank and lower rank, the *rank quotient* of $(E, \mathcal{F})$ is defined by

$$q(E, \mathcal{F}) := \min_{X \subseteq E} \frac{\rho(X)}{r(X)}.$$

The rank quotient directly gives a lower bound for the approximation quality of the Greedy Algorithm by the following theorem. Here, let $\mathrm{G}(E, \mathcal{F}, c)$ be the weight of a solution found by the Greedy Algorithm and let $\mathrm{OPT}(E, \mathcal{F}, c)$ be the weight of an optimum solution for a given weight function $c : E \to \mathbb{R}_+$.

Theorem 3.11 ([26, theorem 13.19, pp. 304f.]) *Let $(E, \mathcal{F})$ be an independence system. Then*

$$q(E, \mathcal{F}) \leq \frac{\mathrm{G}(E, \mathcal{F}, c)}{\mathrm{OPT}(E, \mathcal{F}, c)} \leq 1$$

for all $c : E \to \mathbb{R}_+$. There is a weight function c where the lower bound is attained.

Korte and Hausmann [27, p. 70] have already calculated the rank quotient for the given problem. They have shown that there is no positive lower bound on the approximation quality, i.e.

$$\lim_{n \to \infty} q(E, \mathcal{F})_{K^n} = 0.$$

Their proof uses $F_1 = \{(v_i, v_{i+1} : 1 \leq i \leq n-1\}$, $F_2 = \{(v_i, v_j) : 1 \leq j < i \leq n\}$ and $S = F_1 \cup F_2 \subseteq E$, which is also depicted in figure 3.2. Clearly F_1 and F_2 are bases of S with $n-1$ and $\binom{n}{2}$ elements, respectively. Hence

$$q(E, \mathcal{F})_{K^n} = \min_{X \subseteq E} \frac{\rho(X)}{r(X)} \leq \frac{\rho(S)}{r(S)} = \frac{n-1}{\binom{n}{2}} = \frac{2}{n},$$

which yields the above.

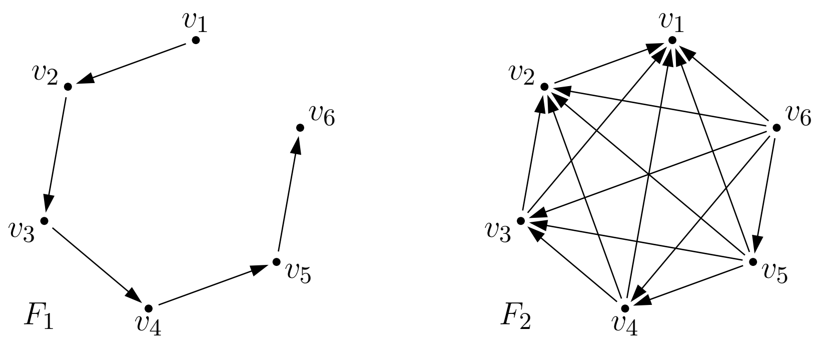

Figure 3.2 Two independent elements of $(E, \mathcal{F})_{K^6}$ showing that the rank quotient can become arbitrary low

Consequently, this means that the proposed Greedy Algorithm can find arbitrary bad solutions compared to the optimum, e.g. for the weight function

$$c(e) := \begin{cases} 1+\varepsilon & \text{for } e \in F_1, \\ 1 & \text{otherwise,} \end{cases}$$

$0 < \varepsilon \ll 1$, the algorithm 3.9 chooses only edges from F_1 and therefore yields $\mathrm{G}(E, \mathcal{F}, c) = (n-1)(1+\varepsilon)$, while the optimum solution, given by e.g. F_2, has weight $\mathrm{OPT}(E, \mathcal{F}, c) = \binom{n}{2}$.

This result shows that the Greedy Algorithm is not a good choice to solve step 3 of algorithm 3.4. Two better approaches are for example the algorithms from Hassin and Rubinstein [22] and from Berger and Shor [19]. They both yield $\left(\frac{1}{2}+\Omega\left(\sqrt{\Delta}^{-1}\right)\right)$-approximations in the unweighted case and have running time of $O(|E|+\Delta^3)$ and $O(|E|\cdot|V|)$, respectively. However, their technique for 2-cycle removal and reintroduction [19, p. 238], on which these algorithms base, yields no approximation guarantee for weighted graphs containing 2-cycles.

In the presented case of a complete digraph, those two algorithms have running time of $O(|V|^3)$ and the gain over the $\frac{1}{2}$-approximation is very small, as Δ is very big. So instead of them the following very easy algorithm is considered. It can be found in a similar form in [19, pp. 237f.].

Algorithm 3.12 (Simple Subdigraph Acyclicity Algorithm)

```
Input:   A complete digraph $K$ with weights $c : E(K) \to \mathbb{R}_+$
Output:  A maximal acyclic subdigraph $H$ of $K$
1. Set $X := \emptyset$ and $W := V(K)$.
2. Choose $v \in V(K)$.
3. If $c(E_K^-(v)) \leq c(E_K^+(v))$, then set $X := X \cup E_K^+(v)$. Otherwise set $X :=$
   $X \cup E_K^-(v)$.
4. Remove $v$ and all incident edges from $K$.
5. If $|V(K)| \geq 1$, then goto 2, otherwise output $H = (W, X)$.
```

The resulting subdigraph H is indeed acyclic. Assume there is a directed cycle C in H. Let v be the first vertex of C chosen by the algorithm in step 2 and let (x, v) and (v, y) be the two edges in C incident with v. Then only one of (x, v) and (v, y) was added in step 3, so C cannot be a cycle [19, theorem 2, p. 238].

Furthermore, the algorithm has n iterations, i.e. steps 2 to 5 are executed n times, with $n = |V(K)|$. Step 4 always yields a complete directed subdigraph of K. Hence, step 3 adds exactly $n - i$ edges to H in the ith iteration. So when the algorithm stops, it holds that

$$E(H) = \sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \binom{n}{2}.$$

An acyclic digraph on n vertices cannot have more than $\binom{n}{2}$ edges, so H is maximal. Hence, the algorithm is correct.

Algorithm 3.12 considers every vertex and every edge once, so it has running time of $O(|V| + |E|) = O(|V|^2)$, which has also been stated in [19, theorem 3, p. 238]. The algorithm has a $\frac{1}{2}$-approximation guarantee, as the following observation shows [19, theorem 1, p. 238].

Observation 3.13 Let K be a complete digraph with a weight function $c : E(K) \to \mathbb{R}_+$ and let H' be a maximum acyclic subdigraph of K. Let H be the maximal acyclic subdigraph of K calculated by algorithm 3.12. Then $c(E(H)) \geq \frac{1}{2}c(E(H'))$.

Proof. The algorithm considers every edge once and always adds at least half of the weight of the deleted edges to the weight of X. □

There is also a trivial algorithm to compute a maximal acyclic digraph from a complete digraph. This trivial algorithm just takes an arbitrary ordering of the vertices and removes all edges not following this ordering. Then it calculates the weight of the so constructed digraph and of its complement digraph, which are both maximal acyclic, and outputs the one with the larger weight.

This trivial algorithm has the same worst-case complexity and approximation guarantee as algorithm 3.12, but the average-case complexity is worse, as the following shows. Both algorithms are very similar with the main difference that algorithm 3.12 decides in step 3 in every iteration, how to orient the edges, without influencing the other iterations. The trivial algorithm only considers those two possible solutions where always $E^+_K(v)$ or always $E^-_K(v)$ is added, so it cannot do better than algorithm 3.12 and is therefore worse on average.

In conclusion, algorithm 3.12 has a good approximation guarantee and acceptable performance, which makes it a good choice for step 3 of algorithm 3.4. With this subalgorithm, algorithm 3.4 has a running time of

$O(|E(G)| + |V(G)|^2)$ and clearly also a $\frac{1}{2}$-approximation guarantee. It has been shown in this section that there is a polynomial time $\frac{1}{2}$-approximation algorithm for finding a large spanning subdigraph of a given FD-graph G which is a transitive s-t-DAG. This is important for later sections.

3.3 Problem Reduction

At first a definition from [6] is needed. Let $G = (V, E)$ be an s-t-DAG. A proper subdigraph G' of G, with $|E(G')| > 1$, is an *autonomous sub-DAG* in G if it is an v-w-DAG for $v, w \in V$ and if for every s-t-path P in G the set of edges $E(P) \cap E(G')$ is either empty or forms a v-w-path in G'. Note that by this definition the two trivial cases where G' is a single edge or $G' = G$ are excluded. The two vertices v and w are called *split vertices* in G.

Bein et al. [6] already pointed out that a given problem for G may be solved independently for each autonomous sub-DAG first. This autonomous sub-DAG can then be replaced by a new edge in G that carries all relevant information. To be able to do so, a weight function $c : E \to \mathbb{R}_+$ is needed, which is provided when considering arbitrary subdigraphs of FD-graphs.

Let $c : E \to \mathbb{R}_+$ be the information content as in section 3.1. Let G' be an autonomous sub-DAG in G with source v and sink w. The *reduction of G' in G*, denoted by G/G', is G but with G' replaced by a new edge $e = (v, w)$ with $c(e) = \sigma(G')$. If the edge $e' = (v, w)$ is already present in $E \setminus E(G')$ then the new edge e also replaces e' and $c(e) = \max\{c(e'), \sigma(G')\}$ is assigned to it instead. Note that G/G' again is an s-t-DAG. If G is transitive, then G/G' is also transitive.

Observation 3.14 Let G be an s-t-DAG containing an autonomous sub-DAG G', together with the information content $c : E(G) \to \mathbb{R}_+$. Then it holds that $\sigma(G) = \sigma(G/G')$.

Proof. Let $e \in E(G/G')$ be the edge replacing G' in G/G'. This e may either be contained in a maximum series-parallel subdigraph H of G/G' or not. If $e \notin E(H)$, then H would also be maximum in G because of the structure of autonomous sub-DAGs. On the other hand, if $e \in E(H)$, then a maximum series-parallel subdigraph of G can be constructed from H by removing e and adding a maximum series-parallel subdigraph of G'. □

Because of observation 3.14, determining $\sigma(G)$ for any weighted s-t-DAG G can be reduced to calculating $\sigma(G')$, where G' is a weighted s-t-DAG without an autonomous sub-DAG.

Observation 3.15 Let G be an s-t-DAG having an edge from s to t. If G' is an autonomous sub-DAG in G with source $v \in V(G)$ and sink $w \in V(G)$, then $\{v, w\}$ is a 2-separator of G. Conversely, if $S \subset V(G)$ is a 2-separator in G and C is a component of $G - S$ not containing s or t, then $G' = G[S \cup V(C)]$ is an autonomous sub-DAG of G.

Proof. Let G' be an autonomous sub-DAG of G as stated above. Assume $\{v, w\}$ is not a separator of G. Then there is a vertex $u \in V(G') \setminus \{v, w\}$ such that there is an s-u-path P_{su} not containing v or an u-t-path P_{ut} not containing w. As G' is a v-w-DAG there is a v-w-path P_u containing u. Let P_v be an s-v-path and let P_w be an w-t-path. Then the path $P_v P_u u P_{ut}$ or the path $P_{su} u P_u P_w$ contradict the definition of G'.

Now let $S = \{v, w\}$ be a 2-separator in G and let C be a component of $G - S$ such that $s, t \notin V(C)$. Let $G' = G[S \cup V(C)]$. It is clear that w.l.o.g it holds that $d^-_{G'}(v) = 0$ and $d^+_{G'}(w) = 0$, otherwise there would be an additional source or sink in G', respectively, which would also be a source or sink in G, or there would be a cycle in G', i.e. in G. Thus G' is a v-w-DAG. In addition, if P is an s-t-path that has edges in common with G', then it necessarily enters G' at v and leaves at w, i.e. it forms a v-w-path in G', otherwise S would not be a separator in G. □

It can be assumed that the edge from s to t is present in any s-t-DAG G because this edge has no influence on whether G is series-parallel or not. If G is transitive then $(s, t) \in E$ anyway. Observation 3.15 implies that the s-t-DAGs without any autonomous sub-DAGs are exactly the 3-connected s-t-DAGs. Hopcroft and Tarjan [28] have shown that it is possible to find all 3-connected components of an arbitrary digraph in linear time based on an idea from planarity testing and a double depth-first search.

4 The Reduction Complexity

Many graph problems that are NP-hard for general s-t-DAGs have been shown to be solvable in polynomial time for series-parallel graphs. This observation motivated Bein et al. [6] to develop the reduction complexity that measures, informally spoken, how nearly series-parallel a general s-t-DAG is. They note that for some NP-hard problems, it is possible to derive algorithms that are exponential only in the reduction complexity of a given s-t-DAG and polynomial in the actual size of the digraph.

This chapter introduces the reduction complexity and outlines its relationship to a minimum vertex cover in a certain auxiliary digraph, as investigated in [6]. Concludingly, the applicability of this approach to the problem presented in chapter 3 is discussed. Within this chapter, let G be an s-t-DAG.

4.1 Node Reductions and the Complexity Graph

The idea of the reduction complexity is the introduction of a third type of reduction in addition to the series and parallel reductions described in section 2.2. Let $v \in V(G)$ have either in-degree or out-degree 1. If v has in-degree 1 then let $e = (u, v)$ be the only edge entering v and let $f_1 = (v, w_1), \dots, f_k = (v, w_k)$ be all edges leaving v. A *node reduction* at v replaces the edges $e, f_1, \dots, f_k$ with the edges $g_1 = (u, w_1), \dots, g_k = (u, w_k)$ and removes v. The case in which $d^+(v) = 1$ is symmetric to the above with $e = (v, w)$ being the only leaving edge, $f_i = (u_i, v)$ being all entering edges and $g_i = (u_i, w)$ being the new edges. Let $G \circ v$ denote the result of a node reduction at v and let $[G]$ denote the digraph that results from G when all possible series and parallel reductions have been applied. The s-t-DAG G is called *irreducible* if $[G] = G$ [6, p. 3].

Now, the *reduction complexity* of G, denoted by $\mu(G)$, is defined as the minimum number k such that there exists a sequence $v_1, \dots, v_k$ with $[\dots[[[G] \circ v_1] \circ v_2] \dots \circ v_k]$ being isomorphic to the digraph $\vec{K}^2$ [6, definition 2.1, p. 3]. This sequence is called *reduction sequence*.

There is a connection between the reduction complexity and a minimum vertex cover of a certain auxiliary digraph, the complexity graph, which makes the calculation of the reduction complexity relatively easy. This connection is discussed in section 4.2, but the necessary definitions and some basic ideas are noted here in advance.

Let $v, w \in V(G)$. The vertex v *dominates* w if every s-w-path in G contains v. Conversely, the vertex w *reverse-dominates* v if every v-t-path in G contains

w. If v dominates w and $v \neq w$, then v is said to *properly dominate* w. Analogously, w *properly reverse-dominates* v if w reverse-dominates v and $v \neq w$.

The *complexity graph* of G, denoted by $C(G)$, is a digraph where $(v, w) \in E(C(G))$ if and only if there exists a v-w-path P in G such that for every vertex $u \in V(P)$, u neither properly dominates w nor properly reverse-dominates v [6, definition 3.1, p. 8]. The set of vertices $V(C(G))$ is defined such that it contains only the end-vertices of edges in $E(C(G))$, thus $C(G)$ has no isolated vertices. An example is shown in figure 4.1.

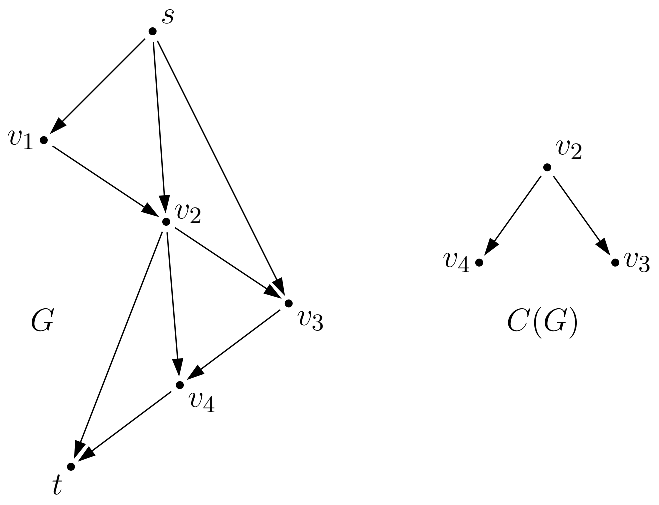

Figure 4.1 A digraph and its complexity graph [6, p. 10]

The main feature of this auxiliary digraph is that it exposes all IGs in G. The first important observation is the following, which is based on the fact that $C([G]) = C(G)$.

Observation 4.1 The complexity graph $C(G)$ is empty if and only if G is series-parallel.

The following lemma provides an alternative definition for $C(G)$. It states that an edge (v, w) is in $C(G)$ if and only if G features a certain path constellation at v and w.

Lemma 4.2 ([6, lemma 3.4, pp. 10–12]) *Let G be an s-t-DAG. An edge (v, w) is in $E(C(G))$ if and only if there exists paths P_1 and P_2 from v to w, P_{vt} from v to t and P_{sw} from s to w in G such that $V(P_1) \cap V(P_{vt}) = \{v\}$ and $V(P_2) \cap V(P_{sw}) = \{w\}$. Then either $P_1 = P_2$ or $V(P_1) \cap V(P_2) = \{v, w\}$, and $P_{sw} \cap P_{vt}$, if not empty, forms a single path.*

A very technical proof of this lemma can be found in [29]. A consequence, which has been documented in [6, 29], of this lemma is the existence of only four different minors of G that force the occurence of an edge (v, w) in $E(C(G))$ and require node reductions at v or w to disappear. These minors are shown in figure 4.2.

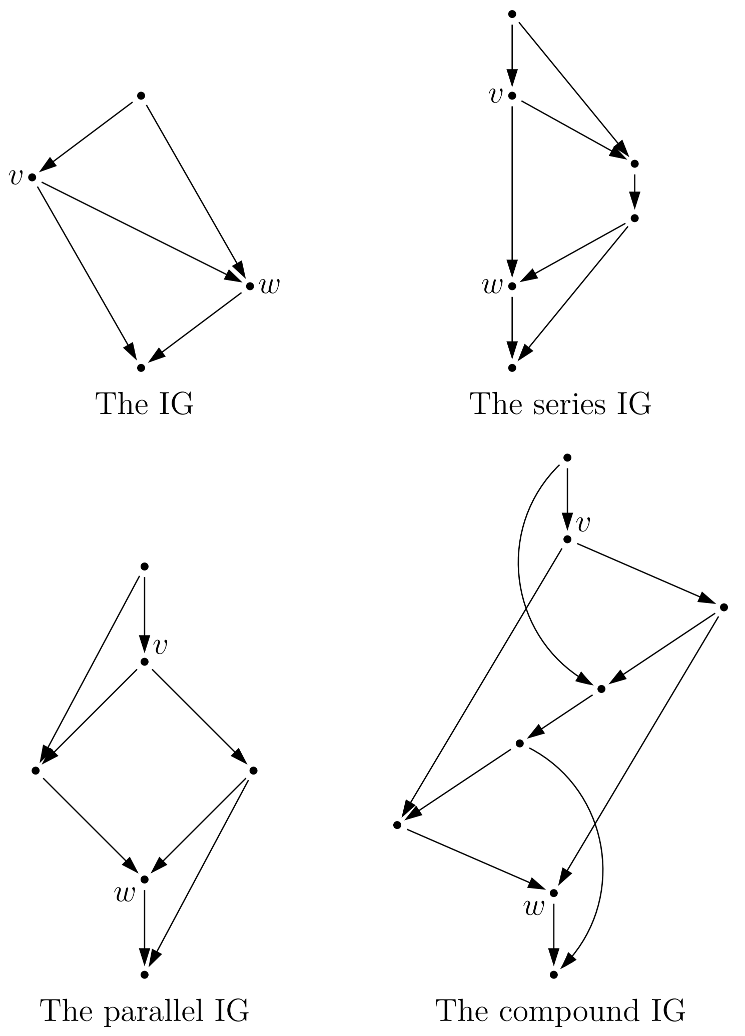

Figure 4.2 The four minors that can appear in G for an edge $(v, w) \in E(C(G))$ [6, p. 11]

4.2 Calculating the Reduction Complexity

In this section it is shown that the reduction complexity of G is equal to the cardinality of a minimum vertex cover in $C(G)$.

Lemma 4.3 ([6, lemma 4.1, p. 12]) *If V' is a minimum vertex cover in $C(G)$, then $|V'| \leq \mu(G)$.*

This lemma can easily be proven by showing that for every edge (v, w) at least v or w must be included in any reduction sequence. Proving the other part, i.e. $|V'| \geq \mu(G)$, is more involved and occupies the remainder of this section.

At first, Bein et al. [6] outline that sources and sinks in $C(G)$ which are not split vertices in G are eligible for node reductions. Recall from section 3.3 that a split vertex is source or sink of an autonomous sub-DAG. Another related result is that a node reduction at an end-vertex of a contractable edge has no influence on the complexity graph, so node reductions have to be avoided. An edge (v, w) is *contractable* if $d^+(v) = d^-(w) = 1$.

The next two lemmata fill the missing gap to theorem 4.6. The first one guarantees that there is always a vertex that can be chosen accordingly to the above findings and the second one shows that a node reduction at such a vertex actually decreases the reduction complexity.

Lemma 4.4 ([6, lemma 4.4, p. 13]) *If G is irreducible and $V' \neq \emptyset$ is a minimum vertex cover of $C(G)$, then there exists $v \in V'$ such that v is source or sink of $C(G)$, not a split vertex of G and not end-vertex of a contractable edge.*

The proof for this lemma initially reduces the search of v to a non-empty component $C(H)$ of $C(G)$ such that H has no autonomous sub-DAG. Then it is proven that there is an edge (s', t') in $C(H)$ such that s' is a source and t' is a sink in $C(H)$, respectively, and clearly one of those two vertices must be in V'.

Lemma 4.5 ([6, lemma 4.5, p. 14]) *If G is irreducible, v is eligible for node reduction in G, i.e. either $d^-(v) = 1$ or $d^+(v) = 1$, and v is not end-vertex of a contractable edge, then $C(G \circ v) \subseteq C(G) - v$.*

Assume that $d^-_G(v) = 1$; for $d^+_G(v)$, the argumentation is symmetric. Here it suffices to show that $(u, w) \in E(C(G \circ v))$ implies $(u, w) \in E(C(G))$ with u being the unique in-neighbor of v. This u cannot be reverse-dominated by v in G, so after inserting v into any u-w-path this path is still valid in terms of the complexity graph definition.

From the above results, theorem 4.6 can now be proven by a simple induction on $|V'|$ because the cardinality of a minimum vertex cover in $C([G \circ v])$ is at most $|V'| - 1$ by lemma 4.5.

Theorem 4.6 ([6, theorem 4.6, p. 14]) *If V' is a minimum vertex cover in $C(G)$, then $\mu(G) = |V'|$.*

The polynomial time algorithm for calculating $\mu(G)$ is also derived in [6, pp. 14–16]. Its main parts are computing $C(G)$ and finding a minimum vertex cover in $C(G)$, but this is not discussed here.

4.3 Importance to the Hierarchy Problem

One of the early ideas for solving the hierarchy problem was the application of an inversed reduction sequence to a $\vec{K}^2$. Starting with the s-t-DAG G, one would apply series, parallel and node reductions, as discussed in the previous sections, and record all those changes and their order. In case of series and parallel reductions, these changes can be represented in form of those subdigraphs of the current s-t-DAG G' that are replaced by single edges in $[G']$.

After that, those changes can be reversed with a slight modification and reapplied to a $\vec{K}^2$ in reverse order. It is obvious how to reverse series and parallel reductions. When reverting a node reduction as defined at the beginning of section 4.1, only one of the edges g_i, say g_1, is replaced by e and f_1, i.e. v is reinserted as a subdividing vertex on g_1. The other edges $g_2, \ldots, g_k$ are left untouched. Let H be the digraph that results after reapplication of all the reversed changes.

The motivation of this approach is that the resulting digraph H is series-parallel because the reverted node reduction is basically a reverted series reduction, and that H is a spanning subdigraph of the transitive closure of G. Thus, H is a spanning subdigraph of G if G is transitive. Since a maximal acyclic subdigraph of an FD-graph is a transitive s-t-DAG, H is a valid candidate for a hierarchy.

However, it is very difficult to find a performance guarantee for this method. Its outcome strongly depends on the choice of the minimum vertex cover in $C(G)$ and it is not clear whether it is a better idea to choose a vertex cover with greater cardinality. Therefore, this strategy has been discarded for now. Several other approaches are discussed in chapter 5.

5 From Transitive s-t-DAGs to Large Intransitive Series-Parallel Digraphs

Recall that objective 3.3 (1) asks for series-parallel subdigraphs of the FD-graph. This chapter deals with this problem in case an acyclic FD-graph is given or an acyclic subdigraph of the FD-graph is created, e.g. by utilizing the ideas from section 3.2.

Let $G = (V, E)$ be a transitive s-t-DAG with the information content $c : E \to \mathbb{R}_+$ from section 3.1; i.e. for a subdigraph $G' = (E', V')$ of G its weight is defined by $c(G') = \sum_{e \in E(TR(G'))} c(e)$. Consequently, only intransitive series-parallel subdigraphs of G have to be considered here for objective 3.3, as transitive edges do not improve the information content of the subdigraph. The issue of finding such a subdigraph with maximum information content in G is discussed in this chapter. In the first two sections, a result for weight functions with certain special properties is discussed, and in section 5.3, the performance of the Greedy Algorithm for this problem is evaluated.

5.1 Equally Weighted Edges

In the easiest case of a weight function, all edges are *equally weighted*, i.e. it holds that $c(e) = 1$ for all $e \in E$. This means that the maximum intransitive series-parallel subdigraph Π in G is the one with the maximum number of edges for a fixed number of vertices, so $|E(\Pi)| - |V(\Pi)|$ has to be maximized. A parallel composition of two s-t-DAGs increases this value by two, while series compositions only increase it by one. Consequently, the digraph aimed at is the intransitive series-parallel digraph created by as many parallel compositions as possible. This is the s-t-DAG Π with $V(\Pi) = V$, in which $N^-_\Pi(v) = \{s\}$ and $N^+_\Pi(v) = \{t\}$ for every vertex $v \in V(\Pi) \setminus \{s, t\}$, and the edge $(s, t) \in E(\Pi)$ if and only if $V(\Pi) = \{s, t\}$. This digraph is depicted in figure 5.1.

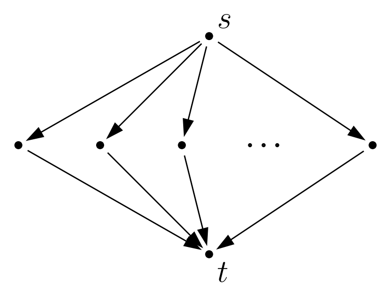

Figure 5.1 The edge-maximum intransitive series-parallel digraph Π

Observation 5.1 Every transitive s-t-DAG G contains Π.

Proof. Every vertex $v \in V \setminus \{s,t\}$ lies on an s-t-path, so there is always an s-v-path and a v-t-path in G. Because of transitivity it follows that $(s,v),(v,t) \in E$, i.e. $\Pi \subset G$. □

The digraph Π clearly has $2(|V|-2)$ edges if $|V| \geq 3$.

Corollary 5.2 *If the edges of G are equally weighted then it holds that $\sigma(G) = c(\Pi) = 2(|V|-2)$ if $|V| \geq 3$, and $\sigma(G) = 1$ otherwise.*

5.2 Major Transitive Weight Functions

If for any three vertices $u,v,w \in V$ with $e_1 = (v,u)$, $e_2 = (u,w)$ and $f = (v,w)$ in E it holds that $c(f) \geq \max\{c(e_1), c(e_2)\}$, then c is called *major transitive*. Note that equally weighted edges are a special case of major transitivity.

Observation 5.3 Let $e \in E$ and let P be a v-w-path in G such that e lies on P. If c is major transitive it holds that $c(e) \leq c((v,w))$.

This observation can easily be shown by induction over the length of P.

Considering major transitive weight functions is a bit more involved than the equally weighted case, but the result is actually the same, as theorem 5.9 shows. However, before proceeding to this theorem some intermediate results are needed.

In the equally weighted case, it was sufficient to show that there are at least as many edges in Π than there are in any intransitive series-parallel subdigraph of G. This idea can be extended to the case of a major transitive information content. The important difference here is that now the information content of the edges has to be considered instead of their mere number.

Let H be an intransitive series-parallel spanning subdigraph of G. First, an auxiliary digraph is introduced, whose vertices are edges in G. Let $A = E(H)$ and let $B = E(\Pi)$. The following considerations require those two sets to be disjoint, but unfortunately they are not. Hence, the elements of B are formally considered as copies of the elements of $E(\Pi)$, although A and B are still treated as subsets of E. Furthermore, let

$$F = \{(a,b) \in A \times B : a \text{ lies on an init}(b)\text{-ter}(b)\text{-path in } H\}.$$

Obviously, the digraph $\mathcal{K} = (A \cup B, F)$ is bipartite with partition classes A and B. By observation 5.3, the following is also clear.

Observation 5.4 If c is major transitive and $(a,b) \in F$ then $c(a) \leq c(b)$.

This means that a matching in $\mathcal{K}$ that matches A yields for every edge in H an edge in Π with same or larger weight, motivating the following.

Proposition 5.5 Let c be major transitive and let $\mathcal{K}$ contain a matching of A. Then it holds that $c(H) \leq c(\Pi)$.

Proof. Let M be a matching of A in $\mathcal{K}$. As H is intransitive and M matches A, it holds that

$$c(H) = \sum_{e \in E(H)} c(e) = \sum_{m \in M} c(\mathrm{init}(m)).$$

Furthermore, because of observation 5.4 and $M \subseteq F$,

$$\sum_{m \in M} c(\mathrm{init}(m)) \leq \sum_{m \in M} c(\mathrm{ter}(m)).$$

With $\mathrm{ter}(m) \in B$ for all $m \in M$ and Π being intransitive this yields

$$c(H) \leq \sum_{m \in M} c(\mathrm{ter}(m)) \leq \sum_{b \in B} c(b) = c(\Pi).$$

□

So it remains to show that there is always a matching in $\mathcal{K}$ that matches A. This can be done by utilizing the well-known result by Hall, sometimes called *marriage theorem*. It is presented here without proof. Three different proofs can be found in [9, pp. 36f.].

Theorem 5.6 (Hall Theorem [9, theorem 2.1.2, p. 36]) *A bipartite digraph with partition classes A and B contains a matching of A if and only if $|N(A')| \geq |A'|$ for all $A' \subseteq A$.*

Let $A' \subseteq A$. A vertex $v \in V$ is called *inner vertex* of A' if there are edges $a_1, a_2 \in A'$, $a_1 \neq a_2$, such that $v = \mathrm{init}(a_1) = \mathrm{ter}(a_2)$. The set of all inner vertices of A' is denoted by $I(A')$. Furthermore, a vertex $v \in V \setminus I(A')$ is a *pre-join vertex* of A' if there are edges $a_1, a_2 \in A'$, $a_1 \neq a_2$, such that $v = \mathrm{init}(a_1)$ and $\mathrm{ter}(a_1) = \mathrm{ter}(a_2)$. On the contrary, v is called a *post-split vertex* of A' if there are edges $a_1, a_2 \in A'$, $a_1 \neq a_2$, such that $v = \mathrm{ter}(a_1)$ and $\mathrm{init}(a_1) = \mathrm{init}(a_2)$. Note that a vertex cannot be pre-join and post-split vertex. The set of all pre-join vertices of A' is denoted by $J^+(A')$, the set of all post-split vertices is $J^-(A')$ and let $J(A') = J^+(A') \cup J^-(A')$. Finally, an edge $a \in A'$ is *separated* in A' if a is not adjacent with any other edge in A'. The set of all separated edges in A' is denoted by $S(A')$.

Proposition 5.7 It holds that $|N_{\mathcal{K}}(A')| \geq 2|I(A')| + |J(A')| + |S(A')|$ for all $A' \subseteq A$.

Proof. Let $A' \subseteq A$. Let N_I be the set of all edges (s, v) and (v, t) with $v \in I(A')$. Note that $s, t \notin I(A')$. Clearly $|N_I| = 2|I(A')|$ and $N_I \subseteq B$. Any edge in A' that ends or starts in v lies on an s-v-path or a v-t-path in H, respectively. So $N_I \subseteq N_{\mathcal{K}}(A')$.

Let N_J be the set of all edges (v, t) with $v \in J^+(A')$ together with all edges (s, w) with $w \in J^-(A')$. Note that $s, t \notin J(A')$ because that would imply a transitive edge in H starting in s or ending in t. Thus, it holds that $|N_J| = |J(A')|$. By definition it is $N_I \cap N_J = \emptyset$. Similar to the considerations above, any edge in A' that starts in $v \in J^+(A')$ lies on a v-t-path in H and any edge ending in $w \in J^-(A')$ lies on an s-w-path in H. So $N_J \subseteq N_{\mathcal{K}}(A')$.

Let N_S be the set containing either the edge $(\mathrm{init}(a), t)$, if $\mathrm{init}(a) \neq s$, or the edge $(s, \mathrm{ter}(a))$ otherwise for all edges $a \in S(A')$; thus, $|N_S| = |S(A')|$. A separated edge in A' is not adjacent to any other edge in A', so $N_S \cap (N_I \cup N_J) = \emptyset$. Any edge $a \in A'$ lies on an s-$\mathrm{ter}(a)$-path as well as on a $\mathrm{init}(a)$-t-path, so $N_S \subseteq N_{\mathcal{K}}(A')$.

By the above consideration it holds that $N_{\mathcal{K}}(A') \supseteq N_I \cup N_J \cup N_S$ and

$$\begin{aligned}|N_{\mathcal{K}}(A')| &\geq |N_I \cup N_J \cup N_S| = |N_I| + |N_J| + |N_S| \\ &= 2|I(A')| + |J(A')| + |S(A')|.\end{aligned}$$

□

Proposition 5.8 It holds that $2|I(A')| + |J(A')| + |S(A')| \geq |A'|$ for all $A' \subseteq A$.

Proof. Let $|A'| = 1$. The only edge in A' is obviously separated, so it holds that $|S(A')| = 1 = |A'|$.

Now assume that the statement is true for all $A'' \subseteq A$ with $1 \leq |A''| < k$ and let $|A'| = k$. Let $a \in A'$ and let $A'' = A' \setminus \{a\}$.

In the first case assume that for all edges $a' \in A''$ it holds that $\mathrm{init}(a) \neq \mathrm{init}(a')$ and $\mathrm{ter}(a) \neq \mathrm{ter}(a')$. If $a \in S(A')$ then $S(A') = S(A'') + 1$ and the statement follows directly. Otherwise there is $a_1 \in A''$ such that $\mathrm{ter}(a_1) = \mathrm{init}(a)$ or there is $a_2 \in A''$ such that $\mathrm{ter}(a) = \mathrm{init}(a_2)$. If a_1 exists then $\mathrm{init}(a) \in I(A') \setminus I(A'')$. Note that either $\mathrm{init}(a)$ may already be a post-split vertex of A'' or a_1 may be a separated edge in A'', but not both. The existence of a_2 has analogous implications, so if either one is in A'' it holds that

$$\begin{aligned}2|I(A')| + |J(A')| + |S(A')| &\geq 2(|I(A'')| + 1) + |J(A'')| + |S(A'')| - 1 \\ &\geq |A''| + 1 = |A'|.\end{aligned}$$

Clearly, if $a_1 \in A''$ and $a_2 \in A''$ it holds that

$$\begin{aligned}2|I(A')| + |J(A')| + |S(A')| &\geq 2(|I(A'')| + 2) + |J(A'')| + |S(A'')| - 2 \\ &\geq |A''| + 2 > |A'|.\end{aligned}$$

For the second case it is assumed that there is an edge $a_1 \in A''$ such that $\mathrm{init}(a) = \mathrm{init}(a_1)$ and that $\mathrm{ter}(a) \neq \mathrm{ter}(a')$ for all edges $a' \in A''$. Clearly, this implies $\mathrm{ter}(a) \notin I(A'')$. If there is an edge $a_2 \in A''$ with $\mathrm{ter}(a) = \mathrm{init}(a_2)$ then $\mathrm{ter}(a) \in I(A')$ and, as above, $|J(A')| + |S(A')| \geq |J(A'')| + |S(A'')| - 1$, yielding the claim. On the other hand, if $\mathrm{ter}(a) \neq \mathrm{init}(a')$ for all edges $a' \in A''$ then $\mathrm{ter}(a) \in J(A') \setminus J(A'')$ because of a_1.

The third case, in which there is no edge $a' \in A''$ with $\mathrm{init}(a) = \mathrm{init}(a')$ but an edge $a_1 \in A''$ with $\mathrm{ter}(a) = \mathrm{ter}(a')$, is symmetric to the second case.

Finally, two edges $a_1, a_2 \in A''$, $a_1 \neq a_2$, with $\mathrm{init}(a) = \mathrm{init}(a_1)$ and $\mathrm{ter}(a) = \mathrm{ter}(a_2)$ are a contradiction to H being series-parallel. $\square$

Indeed, for this proof it is necessary that H is series-parallel, as the example in figure 5.2 shows. There, none of the elements of A' fit into any of the three cases of the proof and it holds that $|A'| = 7$ and $2|I(A')| + |J(A')| + |S(A')| = |J(A')| = 6$.

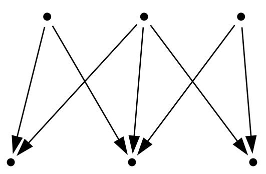

Figure 5.2 An example for A' showing that proposition 5.8 does not hold in general for graphs that are not series-parallel

Those two propositions form the premise for theorem 5.6, so the main result follows easily.

Theorem 5.9 *If the information content c is major transitive, then $\sigma(G) = c(\Pi)$.*

Proof. Clearly, $\sigma(G) \geq c(\Pi)$. Let H be an intransitive series-parallel spanning subdigraph of G with $\sigma(G) = c(H)$. By proposition 5.7, proposition 5.8 and theorem 5.6 there is a matching of A contained in $\mathcal{K}$. So by proposition 5.5 it holds that $c(\Pi) \geq c(H)$. Thus, there must be equality in $\sigma(G) \geq c(\Pi) \geq c(H) = \sigma(G)$. $\square$

It is also possible to construct a matching of A in $\mathcal{K}$ directly by exploiting observation 2.4. This is outlined in the following algorithm.

Algorithm 5.10 (Matching Algorithm)

Input: An intransitive series-parallel digraph H
Output: A matching M of A in $\mathcal{K}$

1. If H is isomorphic to $\vec{K}^2$ then output $M := \{((s,t),(s,t))\}$, otherwise set $M := \emptyset$ and $H' := H$.
2. Choose a subdividing vertex $v \in V(H')$ and let $(u,v),(v,w) \in E(H')$ be its incident edges.
3. Set $M := M \cup \{((u,v),(s,v))\}$ if $(u,v) \in E(H)$.
4. Set $M := M \cup \{((v,w),(v,t))\}$ if $(v,w) \in E(H)$.
5. Remove v and the two incident edges from H'. Now, if there is either $d^+_{H'}(u) = 0$ or $d^-_{H'}(w) = 0$, add the edge (u,w) to H'.
6. If H' is isomorphic to $\vec{K}^2$ then output M, otherwise goto 2.

Clearly, $M \subseteq E(\mathcal{K})$ and M is a matching because v is not chosen again after its deletion in step 5. All edges of H are included in the matching M, either in steps 3 and 4, or in step 1 if H is isomorphic to $\vec{K}^2$. Thus, M matches A. The algorithm terminates after exactly $|V(H)| - 2$ iterations because step 5 removes one vertex and the algorithm stops when there are only two left. Step 5 also asserts that H' remains intransitive and series-parallel during the algorithm, so there always is a subdividing vertex in step 2 by observation 2.4. The condition in steps 3 and 4 is necessary to ensure that only edges from H occur in M, and none of those added in step 5.

Besides step 2, which needs at most $O(|V(H)|)$ evaluations, all steps can be done in constant time. The conditions in steps 3 and 4 can be evaluated in constant time by tagging all edges from $E(H)$ in H' at the beginning. Thus, algorithm 5.10 has a time complexity of $O(|V(H)|^2)$.

This algorithm guarantees the existence of a matching of A in $\mathcal{K}$ and therefore provides an alternative approach to the proof of theorem 5.9 without using Hall's theorem. Using a similar idea it is also possible to prove the theorem directly by the following observation, even without using $\mathcal{K}$ at all.

Observation 5.11 The digraph H is Π if and only if for all subdividing vertices v of H it holds that $v \in N_H(s) \cap N_H(t)$.

Proof. If $|V(H)| = 2$, this is trivial, so let $|V(H)| \geq 3$. Let H' be the digraph resulting from H by removing $N_H(s) \cap N_H(t)$ and all incident edges. If $H' = (\{s,t\},\emptyset)$, then H clearly is Π and all subdividing vertices of H are in $N_H(s) \cap N_H(t)$. On the other hand, if H' is not $(\{s,t\},\emptyset)$, then it must be series-parallel and therefore contain a subdividing vertex by observation 2.4 that is also subdividing vertex of H. □

Hence, if H is not Π, it is possible to find a subdividing vertex $v \in V$ that is not incident to both s and t. Let $N(H) = N_H(s) \cap N_H(t)$. Now there is an easy operation that increases the number of vertices in $N(H)$ while leaving the other properties of H intact. The existence of such an operation motivates the following proof to theorem 5.9.

Proof. Let H be an intransitve series-parallel subdigraph of G with $c(H) = \sigma(G)$ and $|N(H)|$ being maximal. Let H not be Π, otherwise the claim follows directly. Because of observation 5.11 there is a subdividing vertex $v \in V$ of H with $v \notin N(H)$. Let u and w be the unique in- and out-neighbor of v, respectively.

Let H' be the digraph that results from H by removing the edges incident to v, adding the edges (s, v) and (v, t), and adding the edge (u, w) if v lies on the only u-w-path in H. As a result, H' is also intransitive series-parallel.

Clearly, $|N(H)| < |N(H')|$ because v is in $N(H')$ and not in $N(H)$. By observation 5.3 it is $c(H) \leq c(H')$, i.e. $c(H) = c(H') = \sigma(G)$. Thus, H' contradicts the maximality of $N(H)$. $\square$

5.3 The Greedy Approach

For the information content $c : E \to \mathbb{R}_+$ without further restrictions, it is possible to apply the Greedy Algorithm, as introduced in section 3.2, to find an intransitive series-parallel subdigraph of an s-t-DAG. As before, it is necessary to investigate the performance of this algorithm.

Let $\mathcal{F} = \{F \subseteq E : \exists$ intransitive series-parallel spanning subdigraph H of G such that $F \subseteq E(H)\}$. Clearly, $(E, \mathcal{F})$ is an independence system. By this definition, the bases of $(E, \mathcal{F})$ are exactly the intransitive series-parallel spanning subdigraphs of G. So, if H is an optimum solution to the maximization problem $(E, \mathcal{F}, c)$, then it holds that $c(H) = \sigma(G)$.

So the question remains whether the Greedy Algorithm always yields an optimum solution and, if not, whether there is at least an approximation guarantee. Recall that by theorem 3.10 the Greedy Algorithm always obtains optimum solutions if $(E, \mathcal{F})$ is a matroid. It is easy to see that this is not the case in general, and figure 5.3 shows a possible counterexample.

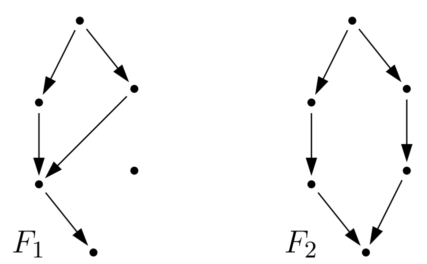

Figure 5.3 Two independent elements of $(E, \mathcal{F})$, $E = F_1 \cup F_2$, violating (M3)

In addition, it is possible to calculate the rank quotient of $(E, \mathcal{F})$. An example can be constructed, which exploits the idea in figure 5.3, where the presence of two edges e_1 and e_2 in an independent element of $(E, \mathcal{F})$ prohibit the addition of any other edge of an arbitrarily long path that could otherwise be added. This example is depicted in figure 5.4 with k being the length of the mentioned path.

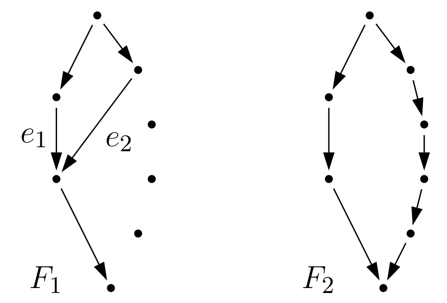

Figure 5.4 An example for two maximal independent sets of $(E, \mathcal{F})$, $E = F_1 \cup F_2$, for $k = 4$ that lead to an arbitrarily low rank quotient

As a result from this example, it holds that

$$q(E, \mathcal{F}) = \min_{X \subseteq E} \frac{\rho(X)}{r(X)} \leq \frac{5}{4+k}.$$

Thus,

$$\lim_{k \to \infty} q(E, \mathcal{F}) = 0,$$

i.e. the Greedy Algorithm has no approximation guarantee and can perform arbitrarily bad. Similar to the example from figure 3.2 in section 3.2, setting $c(e_1) = c(e_2) = 1+\varepsilon$, $0 < \varepsilon \ll 1$, and $c(f) = 1$ for all $f \in E \setminus \{e\}$ demonstrates the issue.

6 Information Content of the Edges

Previous chapters used a general weight function $c : E \to \mathbb{R}_+$ for the edges of an FD-graph, discussing only global properties of c and not the values for individual edges. Doing so made it possible to focus solely on algorithmic approaches that are not bound to a certain weight function. Nonetheless, the construction of a particular weight function, which is the main issue of this chapter, is of great interest for this work. Section 6.1 presents the motivation behind a possible weight function for this problem as introduced in section 3.1, the information content of an edge, and outlines its derivation. Some properties regarding major transitivity are discussed in section 6.2.

6.1 Motivation and Calculation

Within a given relation, there may occur functional dependencies that do not represent the underlying application, but are rather coincidentally. A customer relation might, for example, satisfy an untenable functional dependency between the addresses of the customers and their first names, especially if the relation is very small. To allow the identification of such functional dependencies in the FD-graph, a special edge weight is used, whose calculation is introduced in this section.

The basic idea of the edge weight is that every edge contains a certain amount of information. An edge that represents a functional dependency whose existence is unlikely contains much information, whereas a functional dependency with a high probability of occurence results in a lower information content for the concerned edge. So edges with a high information content can be considered as trustworthy and, when it comes to choosing edges for removal, it is strived for keeping as much of those edges as possible to maximize the summarized information content of the digraph.

The first step is calculating the probability of the occurence of a specific functional dependency in a relation of given size. Let r be a relation over R with $|r| = n$ and let G be the FD-graph of r. Let $\mathcal{A} = \{A_1, \ldots, A_u\} \subseteq R$ and $\mathcal{B} = \{B_1, \ldots, B_v\} \subseteq R$ be two attribute sets. Recall that this implies that the sets in $\mathcal{A} \cup \mathcal{B}$ are pairwise disjoint. Let r satisfy the functional dependency $\mathcal{A} \to \mathcal{B}$ and let $e = (\mathcal{A}, \mathcal{B}) \in E(G)$ be the edge that represents $\mathcal{A} \to \mathcal{B}$. The *FD-probability* of e, denoted by $p(e)$, is defined as the probability that a random relation r' over $\mathcal{A} \cup \mathcal{B}$ with n elements satisfies $\mathcal{A} \to \mathcal{B}$.

A *random* relation r with n elements over a relation schema R is a multiset of n, not necessarily distinct R-tuples, each one chosen independently and uniformly from $\operatorname{dom} R$. Although random relations are multisets, they can still be treated as relations in this context without further adjustments to the definitions from section 2.3.

Let $a = |A_1 \times \ldots \times A_u| = |A_1| \cdot \ldots \cdot |A_u|$ and

$$b = \prod_{S \in \mathcal{B} \setminus \mathcal{A}} |S|.$$

Furthermore, let A and B be two sets with a and b elements, respectively. The FD-probability can then be paraphrased in the following way.

Observation 6.1 The FD-probability $p(e)$ is the probability that when choosing n pairs, such that their first element is from A and their second element is from B, both independently chosen and uniformly distributed random elements, all pairs with the same first element also have the same second element.

Proof. At first let $\mathcal{A}$ and $\mathcal{B}$ be disjoint, thus $b = |B_1 \times \ldots \times B_v|$. Then the statement is clear for $u = v = 1$, since the n pairs are then just corresponding to the elements of r'. For arbitrary values of u and v the elements of r' can still be considered as pairs with the first element from $A_1 \times \ldots \times A_u$ and the second element from $B_1 \times \ldots \times B_v$.

Now, $|\tau \cap A'| = 1$ for every attribute $A' \in R$ and $\tau \in r'$, even if $A' \in \mathcal{A} \cap \mathcal{B}$. So for every pair, the element from $A' \in \mathcal{A} \cap \mathcal{B}$ can only be freely chosen for the first element of that pair, as $A' \in \mathcal{A}$, and not for its second element, thus $|A'|$ must not appear in the product b. This argumentation can be independently used for every attribute in $\mathcal{A} \cap \mathcal{B}$. $\square$

Now consider the function $H(n,k)$ for $n, k \geq 0$, which denotes the number of possibilities to choose n pairs such that there are exactly k different first elements among these pairs. It is clear that $H(n,k) = 0$ if $k > n$ because there cannot be more different first elements than pairs. It holds also true that $H(n,0) = 0$ for $n \neq 0$ because there has to be at least one first element if any pairs are chosen. Finally, set $H(0,0) = 1$ because there is exactly one way to choose 0 pairs.

For the other values of n and k, note that when selecting the nth pair, this pair can either have a new first element or one that is already present among the other $n-1$ pairs. In the latter case there are $k \cdot H(n-1,k)$ possibilities because there are k different first elements and there is only one choice for the second element, as observation 6.1 demands. If, on the other hand, a new first element is chosen, then there are $(a-k+1)b \cdot H(n-1,k-1)$ possibilities. The first element of the new pair can be any element from A that was not already selected by the $n-1$ other pairs and the second element is arbitrary. This yields the recurrence equation

$$H(n,k) = kH(n-1,k) + (a-k+1)bH(n-1,k-1) \tag{6.1}$$

for all $n, k \geq 1$. Now adding up $H(n,k)$ for all possible values for k gives

$$p(e) = \frac{\sum\limits_{k=1}^{a} H(n,k)}{(ab)^n} \tag{6.2}$$

for the FD-probability, where the nominator counts the number of favorable cases following observation 6.1 and where $(ab)^n$ is the number of all possible cases to choose n pairs arbitrarily from $A \times B$.

The limits of the sum in (6.2) can be extended for notational convenience, as the following calculation shows. Because of (6.1) it holds that

$$\begin{aligned} H(n, a+1) &= (a+1)H(n-1,a+1) + (a-(a+1)+1)bH(n-1,a) \\ &= (a+1)H(n-1,a+1) \\ &= (a+1)^n H(0,a+1) = 0 \end{aligned}$$

and $H(n,k)$ with $k > a+1$ always reduces to a sum of such terms and therefore also to 0. Thus (6.2) can be written as

$$p(e) = \frac{\sum\limits_{k\geq 0} H(n,k)}{(ab)^n}. \tag{6.3}$$

A good approach to get a closed equation for $H(n,k)$ is the evaluation of the generating function of $H(n,k)$. In this technique, the values for $H(n,k)$ are considered as coefficients of a formal power series, its *generating function*. A very resourceful book on this topic was written by Wilf [30]. For every $k \geq 0$ let

$$C_k(x) = \sum_{n\geq 0} H(n,k)x^n$$

be the generating function of $H(n,k)$. Now the recurrence equation (6.1) has to be expressed in terms of $C_k(x)$ by multiplying with x^n and summarizing over all values of n for which the recurrence holds. Thus, the left hand side of (6.1) yields for $k \geq 1$

$$\sum_{n\geq 1} H(n,k)x^n = C_k(x) - H(0,k)x^0 = C_k(x).$$

Analogously, the right hand side of (6.1) evaluates to

$$\begin{aligned} &\sum_{n\geq 1}\big(kH(n-1,k) + (a-k+1)bH(n-1,k-1)\big)x^n \\ &= k\sum_{n\geq 0} H(n,k)x^{n+1} + (a-k+1)b\sum_{n\geq 0} H(n,k-1)x^{n+1} \\ &= kC_k(x)x + (a-k+1)bC_{k-1}(x)x \end{aligned}$$

for $k \geq 1$. Combining those two formulas for left and right hand sides gives therefore

$$\begin{aligned} C_k(x) &= kC_k(x)x + (a-k+1)bC_{k-1}(x)x \\ \Leftrightarrow C_k(x) &= \frac{(a-k+1)bx}{1-kx}C_{k-1}(x) \quad \forall k \geq 1, \end{aligned}$$

a recurrence equation for the generating function $C_k(x)$. To solve this, an initial value is needed, which can easily be found in

$$C_0(x) = \sum_{n\geq 0} H(n,0)x^n = 1.$$

Thus, it holds that

$$\begin{aligned} C_k(x) &= \frac{(a-k+1)bx}{1-kx} \cdot \frac{(a-k)bx}{1-(k-1)x} \cdot \ldots \cdot \frac{abx}{1-x} \\ &= a^{\underline{k}} b^k \frac{x^k}{(1-x)(1-2x)\ldots(1-kx)}. \end{aligned} \tag{6.4}$$

Wilf [30, section 1.6] has already shown that the fraction in (6.4) is the generating function of the well-known Stirling numbers of the second kind. But before utilizing Wilf's result, the steps to deduce it are outlined.

The Stirling numbers of the second kind, denoted by $S_{n,k}$, have been discussed in several books [7, 30]. Following those, $S_{n,k}$ is defined as the number of possibilities to partition a set of n elements into k nonempty, pairwise disjoint classes or, equivalently, as the number of different equivalence relations with k equivalence classes on a set with n elements.

It is clear that $S_{n,k} = 0$ for $n < k$ and $S_{n,0} = 0$ for $n > 0$ because a partition of a set of n elements consists of at least one and not more than n classes. In addition, it is $S_{0,0} = 1$. A recurrence equation for $S_{n,k}$ can be deduced by the following consideration [7, p. 16]. W.l.o.g. consider a partition of $\{1, \ldots, n\}$. Now it holds that either $\{n\}$ is one of the classes of the partition or that n is an element of a bigger class. In the first case the other $n-1$ elements are partitioned into $k-1$ classes and there are $S_{n-1,k-1}$ possibilities for that. Removing n in the other case does not reduce the number of classes of the partition, so there are $S_{n-1,k}$ possibilities to partition the other $n-1$ elements and k possiblities to place n in one of the k classes. Altogether this results in

$$S_{n,k} = S_{n-1,k-1} + kS_{n-1,k} \quad \forall n, k \geq 1. \tag{6.5}$$

As before, an equation for the generating function of $S_{n,k}$,

$$B_k(x) = \sum_{n\geq 0} S_{n,k}x^n \quad (k \geq 0),$$

can be found by multiplying (6.5) with x^n and summing over $n \geq 1$, i.e.

$$\sum_{n\geq 1} S_{n,k}x^n = \sum_{n\geq 1} S_{n-1,k-1}x^n + k\sum_{n\geq 1} S_{n-1,k}x^n$$

for $k \geq 1$. This, in turn, gives

$$\begin{aligned}
&B_k(x) = xB_{k-1}(x) + kxB_k(x) \quad (k \geq 1)\\
\Leftrightarrow\ &B_k(x) = \frac{x}{1-kx}B_{k-1}(x) \quad (k \geq 1)\\
\Leftrightarrow\ &B_k(x) = \sum_{n\geq 0} S_{n,k}x^n = \frac{x^k}{(1-x)(1-2x)\ldots(1-kx)} \quad (k \geq 0) \qquad (6.6)
\end{aligned}$$

because of $B_0(x) = 1$.

Now incorporating (6.6) into the generating function of $H(n,k)$ in (6.4) leads to

$$\begin{aligned}
&\sum_{n\geq 0} H(n,k)x^n = C_k(x) = a^{\underline{k}}b^k \sum_{n\geq 0} S_{n,k}x^n = \sum_{n\geq 0} a^{\underline{k}}b^k S_{n,k}x^n\\
\Leftrightarrow\ &H(n,k) = a^{\underline{k}}b^k S_{n,k}. \qquad (6.7)
\end{aligned}$$

Note that Wilf [30, p. 19] also calculated an explicit formula for the Stirling numbers of the second kind by evaluating the partial fraction expansion of (6.6), which has the form

$$\frac{1}{(1-x)(1-2x)\ldots(1-kx)} = \sum_{i=1}^{k} \frac{\alpha_i}{1-ix}.$$

Here, the α's can be isolated by choosing a fixed r, $1 \leq r \leq k$, multiplying with $1 - rx$ and evaluating at $x = \frac{1}{r}$. After simplification this gives

$$\alpha_i = (-1)^{k-r}\frac{r^{k-1}}{(r-1)!(k-r)!}$$

and the calculation in [30], with $[x^n]f(x)$ denoting the coefficient of x^n in the power series expansion of $f(x)$, shows that

$$\begin{aligned}
S_{n,k} &= [x^n]B_k(x) = [x^n]\frac{x^k}{(1-x)(1-2x)\ldots(1-kx)}\\
&= [x^{n-k}]\frac{1}{(1-x)(1-2x)\ldots(1-kx)}\\
&= \sum_{r=1}^{k} \alpha_r [x^{n-k}]\frac{1}{1-rx} = \sum_{r=1}^{k} \alpha_r r^{n-k}\\
&= \sum_{r=1}^{k} (-1)^{k-r}\frac{r^n}{r!(k-r)!}
\end{aligned}$$

for $n, k \geq 0$. Together with (6.7), this yields

$$H(n,k) = a^{\underline{k}} b^k \sum_{r=1}^{k} (-1)^{k-r} \frac{r^n}{r!(k-r)!} \tag{6.8}$$

as an explicite, closed formula for $H(n,k)$.

There is also a combinatorial explanation for $H(n,k)$ which leads directly to (6.7). Each of the n chosen pairs has one of k first elements. This is equivalent to saying that the n pairs are distributed over k classes of different first elements, and it has already been discussed that there are $S_{n,k}$ possibilities doing so. Furthermore, after choosing an element from A as the first element of a pair there is one possibility less for the other pairs with a different first element, so there are $a^{\underline{k}}$ possibilities of selecting all k different first elements of the pairs. On the other hand, any element from B can be used as second element in a pair with a new first element, thus yielding b^k possibilities for the second element of all pairs. Combining these produces $H(n,k) = a^{\underline{k}} b^k S_{n,k}$, the already known equation.

Now inserting the above result (6.7) in the previous formula (6.3) for $p(e)$ gives

$$p(e) = \frac{\sum_{k \geq 0} a^{\underline{k}} b^k S_{n,k}}{(ab)^n}, \tag{6.9}$$

a closed formula for $p(e)$, and the numbers $S_{n,k}$ may be calculated as in (6.8). Here it is interesting to note that $a^{\underline{k}} b^k S_{n,k}$ vanishes for $k > \min\{a, n\}$, so the range of the sum can indeed be $k \geq 0$.

Assume that $e = (\mathcal{A}, \mathcal{B})$ represents a trivial functional dependency $\mathcal{A} \to \mathcal{B}$, i.e. $\mathcal{B} \setminus \mathcal{A} = \emptyset$. Then the product b is reduced to the neutral element of the multiplication, $b = 1$, and thus

$$p(e) = \frac{\sum_{k \geq 0} a^{\underline{k}} S_{n,k}}{a^n}.$$

Together with

$$a^n = \sum_{k \geq 0} a^{\underline{k}} S_{n,k}, \tag{6.10}$$

which is a known relation between power, falling factorial and Stirling numbers [7, p. 9], this yields $p(e) = 1$. On the other hand, if $b > 1$ then by (6.10) it holds that

$$a^n b^n = \sum_{k \geq 0} a^{\underline{k}} b^n S_{n,k} > \sum_{k \geq 0} a^{\underline{k}} b^k S_{n,k},$$

thus, $p(e) < 1$ in this case.

After finding $p(e)$ in (6.9), the weight of e is defined as $c(e) = -\log p(e)$. Taking the logarithm follows the argumentation of Shannon [31, p. 1] to make the usage of c more convenient as a measure of information. This definition yields a high information content for edges representing functional dependencies with low probability and vice versa, which was an important requirement in the introduction of this weight function. Note that because of the above consideration, $c(e) = 0$ if and only if e represents a trivial functional dependency.

6.2 Major Transitivity

The results in section 5.2 ask for examining in which cases the information content c is a major transitive weight function. Unfortunately, the information content, as calculated in the previous section, has a rather complex analytical structure, which is why this question is hard to answer in a general way.

This section shows that the information content can be major transitive, but also that major transitivity is only a special case that does not occur systematically. This is important to note because this weight function would be useless if the digraph Π was a common optimum hierarchy, as this is just not the case in many complex real world examples.

Let G be a maximal acyclic subdigraph of the FD-graph of the relation r over R with $|r| = n$ as before. Let $\mathcal{A}, \mathcal{B}, C \subseteq R$ be attribute sets, i.e. $\mathcal{A}, \mathcal{B}, \mathcal{C} \in V(G)$, such that $e_1 = (\mathcal{A}, \mathcal{B})$, $e_2 = (\mathcal{B}, \mathcal{C})$ and $e_3 = (\mathcal{A}, \mathcal{C})$ are edges in G. Furthermore, let

$$a_1 = a_3 = \prod_{S \in \mathcal{A}} |S|, \quad a_2 = \prod_{S \in \mathcal{B}} |S|,$$
$$b_1 = \prod_{S \in \mathcal{B} \setminus \mathcal{A}} |S|, \quad b_2 = \prod_{S \in \mathcal{C} \setminus \mathcal{B}} |S|, \quad b_3 = \prod_{S \in \mathcal{C} \setminus \mathcal{A}} |S|,$$

i.e. a_i and b_i take the places of a and b in (6.9) when calculating $p(e_i)$ for $i \in \{1, 2, 3\}$.

The cardinalities of the attributes in $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ are independent from each other, which allows almost arbitrary combinations of the above values. Assume that

$$b_1 \leq b_3, \quad b_2 \leq b_3 \tag{6.11}$$

and for all $0 \leq k \leq n$

$$\begin{aligned} & a_2^n \binom{a_3}{k} \leq a_3^n \binom{a_2}{k} \\ \Leftrightarrow \quad & a_2^n \binom{a_3}{k} k! \leq a_3^n \binom{a_2}{k} k! \\ \Leftrightarrow \quad & \frac{a_3^{\underline{k}}}{a_3^n} \leq \frac{a_2^{\underline{k}}}{a_2^n}. \end{aligned} \tag{6.12}$$

It is not enough to impose a simple restriction on the relationship between a_2 and a_3, because either one of the sides in (6.12) may be greater depending on n and k.

Combining the inequalities (6.12) and (6.11) yields

$$\begin{aligned} \frac{a_3^k b_3^k}{a_3^n b_3^n} &\leq \frac{a_2^k b_2^k}{a_2^n b_2^n} \quad \forall 0 \leq k \leq n \\ \Rightarrow \quad \frac{\sum\limits_{k\geq 0} a_3^k b_3^k S_{n,k}}{a_3^n b_3^n} &\leq \frac{\sum\limits_{k\geq 0} a_2^k b_2^k S_{n,k}}{a_2^n b_2^n}, \end{aligned}$$

i.e. $p(e_3) \leq p(e_2)$ and therefore $c(e_3) \geq c(e_2)$.

Furthermore, it is

$$\frac{a_3^k b_3^k}{a_3^n b_3^n} \leq \frac{a_1^k b_2^k}{a_1^n b_1^n} \quad \forall 0 \leq k \leq n$$

because of $a_1 = a_3$ and (6.11). Analogously to the above, it holds that $c(e_3) \geq c(e_1)$.

So it can be seen that under the constraints (6.11) and (6.12) for all $\mathcal{A}, \mathcal{B}, \mathcal{C} \in V(G)$ with $e_1, e_2, e_3 \in E(G)$ as above, the information content c is indeed major transitive.

7 Conclusions and Open Problems

This work presented a mathematical method to deal formally with the problem of reconstructing attribute hierarchies from database relations, the hierarchy problem. The suggested approach consists of three main steps. At first, the transitive FD-graph $G = (V, E)$ had to be constructed with an appropriate edge weight function that captures the structure of the relation well. Second, a strategy was proposed to remove all cycles from G, which yields a transitive two-terminal acyclic subdigraph of G. This subdigraph was then the basis in the third step to find a large intransitive series-parallel subdigraph of G.

Despite the title of this work, in section 3.1 the consideration of subdigraphs of G instead of minors was propagated. The important reasons for this specialization are that edge contractions in an FD-graph are, because of its special semantics, very similar to certain deletion operations and that it is not clear how to generate weights for the new edges in a minor. Nonetheless it might be possible to overcome the hurdle of generating new edge weights from known ones, which would possibly allow different approaches.

A related question is whether the information content introduced in chapter 6 is a proper model for the desired structure at all, i.e. whether maximum intransitive series-parallel subdigraphs of the FD-graph do indeed represent ideal attribute hierarchies from a database design perspective. The design of the information content was based on the idea that those functional dependencies that are unlikely to occur by accident are more interesting for the construction of a hierarchy. It was out of scope for this work to apply the presented techniques to real world examples, which tend to be extremely large, but such an evaluation is likely to show important properties for a good weight function, which may lead to improvements in the design of the information content.

For the second step, different efficient acyclicity algorithms were discussed in section 3.2. It was shown that they all yield solutions with at least $\frac{1}{2}$ of the optimum weight, which is the best constant lower bound on the approximation quality for all known polynomial acyclicity algorithms. It could also be interesting to use exact acyclicity algorithms instead if there are only small cycles, to reduce the error introduced by this intermediate step. A lot of big cycles would then slow down the whole process considerably, of course.

A different idea to turn the FD-graph G into an s-t-DAG is motivated by the fact that attributes within one hierarchy level are likely to form cyclic functional dependencies. So interpreting cycles as indication that the involved attributes could be grouped together, i.e. that all vertices of a cycle in G that are proper subsets of any other vertex in the same cycle could be removed, is reasonable. Note that there would remain exactly one vertex in any cycle of G because $(v, w), (w, v) \in E$ implies $(v, v \cup w), (w, v \cup w) \in E$, i.e. $v \cup w$ is part of every cycle that contains both v and w. This is also an interesting starting point for an approach to property (2) of the objectives 3.1, 3.2 and 3.3, i.e. the requirement that the set of vertices of a hierarchy is a partition of the set of attributes. However, this necessarily involves the deletion of vertices, which is not favored by the information content $c : E \to \mathbb{R}_+$. Nonetheless, it might be possible to integrate such cycle reductions in a natural way by modifying the weight function, e.g. by introducing negative values for certain or all edge weights in cycles.

Furthermore, it may also be interesting to reconsider the ideas and methods from section 3.3 and chapter 5 if the given graph is cyclic. It seems to be possible to keep the general ideas intact while refining them for the application on cyclic graphs. If this is true, then the second step can be rendered obsolete.

In section 3.3, a possibility to reduce the problem was discussed. The result was that it is only necessary to be able to find maximum intransitive series-parallel subdigraphs in 3-connected s-t-DAGs. The restriction to this class of digraphs did not lead to better theoretical results in this work, but further investigation is required. However, utilizing the reduction of autonomous sub-DAGs in the given s-t-DAG benefits the running time of any applied non-linear algorithm because the size of the individual input digraphs is reduced.

In case of a major transitive weight function, it was shown in section 5.2 that optimum solutions for the third step can be found easily in form of a certain digraph Π that is always a subdigraph of G. Essentially, two approaches were presented. One followed the idea that for every edge in an intransitive series-parallel subdigraph of G there was an appropriate edge in Π with larger weight to replace it, which was verified by proving the existence of a matching in a certain auxiliary digraph utilizing Hall's marriage theorem. The other, more concise approach exploited a basic property of series-parallel digraphs and presented a local operation, whose repeated application turns any intransitive series-parallel subdigraph of G into Π in at most $O(|V|)$ steps while not decreasing its weight. Section 5.1 demonstrated the same result for the simpler case of equally weighted edges, which is a special case of major transitivity.

An important implication is that in case of a major transitive weight function for G, these algorithms provide an optimum solution to the hierarchy problem if the original FD-graph is acyclic. Otherwise the polynomial-time acyclicity algorithm may introduce a certain error in general. Section 5.3 dealt with general weight functions and discussed the application of the Greedy Algorithm to the third step. However, it was shown that this algorithm can yield arbitrarily bad results, just as for the problem of finding a maximum acyclic subdigraph in section 3.2. This and the fact that no further theoretical results were found might be indications that this problem is NP-hard, but unfortunately it was not possible to classify its complexity within this work.

In chapter 4, a paper about a related issue was discussed which was believed to yield another approach to the third step. That paper introduced the reduction complexity to measure the deviation of an arbitrary *s*-*t*-DAG from being series-parallel. However, it became clear that the introduced method, as described in section 4.3, based on their work was to difficult to analyze, and was dismissed in favor of the other ideas.

8 References

[1] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.* (John Wiley & Sons, New York, 2002), 2nd ed.

[2] E. Malinowski and E. Zimányi, *Hierarchies in a Multidimensional Model: From Conceptual Modeling to Logical Representation*, Data & Knowledge Engineering **59** (2006), no. 2, pp. 348–377.

[3] E. F. Codd, *The Relational Model for Database Management: Version 2.* (Addison-Wesley, Reading, MA, 1990).

[4] S. Rizzi, A. Abelló, J. Lechtenbörger, and J. Trujillo *Research in Data Warehouse Modeling and Design: Dead or Alive?*, in *Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP* (ACM, Arlington, VA, 2006), pp. 3–10.

[5] M. Levene and G. Loizou, *Why Is the Snowflake Schema a Good Data Warehouse Design?*, Information Systems **28** (2003), no. 3, pp. 225–240.

[6] W. W. Bein, J. Kamburowski, and M. F. M. Stallmann, *Optimal Reduction of Two-Terminal Directed Acyclic Graphs*, SIAM Journal on Computing **21** (1992), pp. 1112–1129.

[7] M. Aigner, *Diskrete Mathematik.* (Vieweg, Wiesbaden, 2006), 6th ed.

[8] J. Bang-Jensen and G. Z. Gutin, *Digraphs. Theory, Algorithms and Applications.* (Springer, London, 2009), 2nd ed.

[9] R. Diestel, *Graph Theory.* (Springer, New York, 2005), 3rd ed.

[10] K. Thulasiraman and M. N. S. Swamy, *Graphs: Theory and Algorithms.* (John Wiley & Sons, New York, 1992).

[11] J. Valdes, R. E. Tarjan, and E. L. Lawler, *The Recognition of Series Parallel Digraphs*, SIAM Journal on Computing **11** (1982), no. 2, pp. 298–313.

[12] R. J. Duffin, *Topology of Series-Parallel Networks*, Journal of Mathematical Analysis and Applications **10** (1965), no. 2, pp. 303–318.

[13] G. Lausen, Relational Databases in RDF: Keys and Foreign Keys, in *Semantic Web, Ontologies and Databases*, number 5005 in Lecture Notes in Computer Science (Springer, Berlin), pp. 43–56.

[14] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson, *A Foundation for Capturing and Querying Complex Multidimensional Data*, Information Systems **26** (2001), no. 5, pp. 383–423.

[15] P. Vassiliadis and T. Sellis, *A Survey of Logical Models for OLAP Databases*, ACM SIGMOD Record **28** (1999), no. 4, pp. 64–69.

[16] A. Abelló, J. Samos, and F. Saltor *Understanding Analysis Dimensions in a Multidimensional Object-Oriented Model*, in *Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses* (2001), *DMDW '01*, pp. 4-1–4-9.

[17] A. Abelló, J. Samos, and F. Saltor, *YAM2: A Multidimensional Conceptual Model Extending UML*, Information Systems **31** (2006), no. 6, pp. 541–567.

[18] J. Lechtenbörger and G. Vossen, *Multidimensional Normal Forms for Data Warehouse Design*, Information Systems **28** (2003), no. 5, pp. 415–434.

[19] B. Berger and P. W. Shor *Approximation Algorithms for the Maximum Acyclic Subgraph Problem*, in *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990), *SODA '90*, pp. 236–243.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness.* (Freeman, New York, NY, 1979).

[21] V. Guruswami, R. Manokaran, and P. Raghavendra *Beating the Random Ordering is Hard: Inapproximability of Maximum Acyclic Subgraph*, in *49th Annual IEEE Symposium on Foundations of Computer Science* (Philadelphia, PA, 2008), *FOCS '08*, pp. 573–582.

[22] R. Hassin and S. Rubinstein, *Approximations for the Maximum Acyclic Subgraph Problem*, Information Processing Letters **51** (1994), no. 3, pp. 133–140.

[23] J. Leung and J. Lee, *More Facets from Fences for Linear Ordering and Acyclic Subgraph Polytopes*, Discrete Applied Mathematics **50** (1994), no. 2, pp. 185–200.

[24] J. Oxley, *Matroid Theory*, volume 21 of *Oxford Graduate Texts in Mathematics.* (Oxford University Press, 2011), 2nd ed.

[25] S. Even, *Graph Algorithms.* (Cambridge University Press, 2011), 2nd ed.

[26] B. H. Korte and J. Vygen, *Combinatorial Optimization. Theory and Algorithms*, volume 21 of *Algorithms and Combinatorics.* (Springer, Berlin, 2006), 3rd ed.

[27] B. Korte and D. Hausmann, An Analysis of the Greedy Heuristic for Independence Systems, in *Algorithmic Aspects of Combinatorics*, edited by B. Alspach, P. Hell, and D. J. Miller, number 2 in Annals of Discrete Mathematics (Elsevier), pp. 65–74.

[28] J. E. Hopcroft and R. E. Tarjan, *Dividing a Graph into Triconnected Components*, SIAM Journal on Computing **2** (1973), no. 3, pp. 135–158.

[29] W. W. Bein, J. Kamburowski, and M. F. M. Stallmann, *Alternate Characterizations of the Complexity Graph* Technical Report 91-21, Department of Computer Science, North Carolina State University (1991).

[30] H. S. Wilf, *generatingfunctionology.* (Academic Press, Philadelphia, PA, 1994), 2nd ed.

[31] C. E. Shannon, *A Mathematical Theory of Cummunication*, Bell System Technical Journal **27** (1948), pp. 379–423, 623–656.